



EXAMINATIONS — 2009

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.

Java Documentation will be provided with the exam script.

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

Questions

| | Marks |
|-------------------------------------|--------------|
| 1. Understanding Java | [63] |
| 2. Files | [20] |
| 3. Arrays of Objects | [30] |
| 4. 2D Arrays | [30] |
| 5. Designing with Interface Classes | [30] |
| 6. Recursion | [7] |

SPARE PAGE FOR EXTRA ANSWERS

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

Question 1. Understanding Java

[63 marks]

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

```
String x = "Ex";
String y = "Why";
String temp = x;
x = y;
y = temp;
x = x + x;
String z = y + "Zed" + x;
System.out.println("x=" + x);
System.out.println(" y=" + y);
System.out.println(" z=" + z);
```

```
x=WhyWhy
y=Ex
z=ExZedWhyWhy
```

(b) [6 marks] Consider the following condition method (note the “ifs” and “elses” carefully) :

```
public int condition( int x, int y) {
    int ans = 0;
    if (x > y)    { ans = ans + x; }
    else if (x < y) { ans = ans + y; }
    if (x > 2)    { ans = ans + 20; }
    else if (y < 9) { ans = ans + 40; }
    return ans;
}
```

What would the following calls to condition return?

```
condition(1, 10) ==> 10
condition(5, 2) ==> 25
condition(7, 7) ==> 20
```

(Question 1 continued on next page)

(Question 1 continued)

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

```
for ( int j=0; j<3; j++){
    for ( int k=j+1; k<=3; k++){
        System.out.printf(" (%d, %d) ", j, k);
    }
}
System.out.println("Done");
```

(0, 1) (0, 2) (0, 3) (1, 2) (1, 3) (2, 3) Done

(d) [5 marks] What will the following fragment of Java print out if the user enters the three words Queue, On, and Yes in response to the prompts. (Note the println and print carefully.)

```
Scanner sc = new Scanner(System.in);
String word = "No";
while ( ! word.equals("Yes") ){
    System.out.println("Word = " + word);
    System.out.print("Answer: ");
    word = sc.next();
    System.out.println("length = " + word.length());
}
System.out.println("Done");
```

```
Word = No
Answer: Queue
length = 5
Word = Queue
Answer: On
length = 2
Word = On
Answer: Yes
length = 3
Done
```

(Question 1 continued on next page)

(Question 1 continued)

(e) [6 marks] Suppose the variable `numbers` is declared and initialised as follows:

```
int [ ] numbers = new int [ ] {5, 2, 4, 0, 4, 3};
```

numbers:

| | | | | | |
|---|---|---|---|---|---|
| 5 | 2 | 4 | 0 | 4 | 3 |
| 0 | 1 | 2 | 3 | 4 | 5 |

What will the following code fragment print out?

```
for( int j=numbers.length-1; j >= 0; j--){
    System.out.print(numbers[j]+ " ");
}
System.out.println ();
for ( int k=0; k<numbers.length; k++){
    int index = numbers[k];
    System.out.printf (" (%d -> %d) ", k, numbers[index]);
}
System.out.println ();
```

```
3 4 0 4 2 5
(0 -> 3) (1 -> 4) (2 -> 4) (3 -> 5) (4 -> 4) (5 -> 0)
```

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

```
int [ ] numbers = new int [ ] {5, 2, 4, 0, 4, 3};
```

Show the contents of `numbers` after the following `change` method is called on `numbers`:
`change(numbers)`.

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

numbers:

| | | | | | |
|---|---|----|----|----|----|
| 5 | 7 | 11 | 11 | 15 | 18 |
| 0 | 1 | 2 | 3 | 4 | 5 |

(Question 1 continued on next page)

(Question 1 continued)

The ChessPlayer class on the facing page defines ChessPlayer objects, which have two fields to store their name and rating, and two methods. The class also contains a test method.

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

```
A: Jim:(0)
B: John:(0)
C: Jim:(4)
D: John:(2)
E: Snr Jim:(5)
F: John:(7)
```

(h) [6 marks] Write an additional method for the ChessPlayer class called penalise with no parameters which halves the rating of a player if they do not have the title "Snr" at the start of their name.

```
public void penalise(){
    if ( !this.name.startsWith("Snr")){
        this.rating = this.rating / 2;
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

```
public class ChessPlayer{
    private String name;
    private int rating;

    public ChessPlayer(String name){
        this.name = name;
        this.rating = 0;
    }

    public void win (int points){
        this.rating = this.rating + points;
        if ( this.rating == 5) {
            this.name = "Snr " + this.name;
        }
    }

    public String toString(){
        return (this.name + " : ( " + this.rating + " ) ");
    }

    public static void test(){
        ChessPlayer p1 = new ChessPlayer("Jim");
        ChessPlayer p2 = new ChessPlayer("John");

        System.out.println("A: " + p1.toString());
        System.out.println("B: " + p2.toString());

        p1.win(4);
        p2.win(2);

        System.out.println("C: " + p1.toString());
        System.out.println("D: " + p2.toString());

        p1.win(1);
        p2.win(5);

        System.out.println("E: " + p1.toString());
        System.out.println("F: " + p2.toString());
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

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

```
Justin
*Jeremy
*Julia
Jude
```

What will the following `printFile` method print out?

```
public void printFile (){
    try{
        Scanner scan = new Scanner (new File("names.txt"));
        int num = 0;
        while ( scan.hasNext() ){
            String str = scan.next();
            if ( str.startsWith(" * ") ) {
                System.out.println( str.length ());
                num = num + str.length();
            }
            else {
                System.out.println( str );
            }
        }
        System.out.println("Num = " + num);
        scan.close();
    }
    catch(Exception e){System.out.println("File reading failed");}
}
```

```
Justin
7
6
Jude
Num = 13
```

(Question 1 continued on next page)

(Question 1 continued)

(j) [6 marks] Complete the following `mean` method. `mean` has one parameter – an array of doubles – and should return the average of the values in the array. You may assume that the length of the array is greater than 0.

```

public double mean(double [] data){

    double total = 0;
    for (int i=0; i<data.length; i++){
        total = total + data[i];
    }
    return total / data.length;
//OR (alternative loop :)
    for (double n : data){
        total += n;
    }

}

```

(k) [8 marks] Complete the following definition of an `Instrument` class. `Instrument` objects should have two fields called `category` and `lowest`. `category` should hold a `String` specifying what kind of instrument it is, and `lowest` should hold a number specifying the frequency of the lowest note the instrument can play (eg 342.7).

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

```

public class Instrument {
    private String category;
    private double lowest;

    public Instrument(String c, double l){
        this.category = c;
        this.lowest = l;
    }

    public String toString(){
        return this.category + " (" + this.lowest + ") ";
    }

}

```

Question 2. Files

[20 marks]

Suppose the file `invoice.txt` contains data about a supermarket order. Each line contains the unit price (in cents) of an item, the quantity, and the name of the item. If the quantity is 1, the quantity column is left blank. For example, `invoice.txt` might contain:

```
275    Cabbage
145 10 Carrot
540 2  Chocolate Ice Cream
340    Tofu
300 3  White Bread
```

(a) [4 marks] What will the following print method print out if called with the statement `print("invoice.txt");`

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

```
Carrot @ 10
Chocolate Ice Cream @ 2
White Bread @ 3
```

(Question 2 continued on next page)

(Question 2 continued)

(b) [6 marks] Complete the following `printTotalPrice` method so it reads data from a file in the format described above and prints out the item names and total price in dollars of all items. (The total price is the unit price \times quantity.) On the example file above, it should print

| | |
|---------------------|---------|
| Cabbage | \$2.75 |
| Carrot | \$14.50 |
| Chocolate Ice Cream | \$10.80 |
| Tofu | \$3.40 |
| White Bread | \$9.00 |

You do not need to make the items and prices line up.

```

public void printTotalPrice (String fname){
    try{
        Scanner fileScan = new Scanner(new File(fname));
        while(fileScan.hasNext()) {
            double price = fileScan.nextInt()/100.0;
            if (fileScan.hasNextInt()){
                price = price * fileScan.nextInt ();
            }
            String item = fileScan.nextLine ();
            System.out.printf ("%s \t$%4.2f\n", item, price);
        }

        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) [10 marks] Suppose the file `invoice1.txt` contained the supermarket order in an alternative format in which the item is first on each line, followed by the unit price and quantity (if the quantity is greater than 1):

```
Cabbage           275
Carrot            145 10
Chocolate Ice Cream 540 2
Tofu              340
White Bread       300 3
```

Complete the following `lowQuantities` method which should read the data from a file in this format, and print out the names of the items with a quantity of 3 or less (including items with a quantity of one).

For example, calling `lowQuantities("invoice1.txt")` should print:

```
Cabbage
Chocolate Ice Cream
Tofu
White Bread
```

You may assume that the names of items consist only of letters and spaces.

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

        while(fileScan.hasNext()) {
            String item = fileScan.next();
            while (!fileScan.hasNextInt()){
                item = item + " " + fileScan.next();
            }
            int price = fileScan.nextInt();
            int quantity = 1;
            if (fileScan.hasNextInt()){
                quantity = fileScan.nextInt();
            }
            if (quantity <= 3) {
                System.out.println(item);
            }
        }
    }
    // OR
    :
    int price = fileScan.nextInt();
    if (!fileScan.hasNextInt() || fileScan.nextInt()<=3){
        System.out.println(item);
    }
    fileScan.close();
}
catch(IOException e){System.out.println("File reading failed: "+e);}
}
```

Question 3. Arrays of Objects

[30 marks]

This question concerns a program that lets an ESL (English as a Second Language) teacher keep track of the words that a student needs to work on. A student may be having trouble with either the spelling or the pronunciation of a word (or both). The program uses an array of `WordToLearn` objects to store all the words the student is currently learning. The user can add a new word, record that the student has learned the word, and print lists of all the words the student needs to learn the spelling or pronunciation of.

The `WordToLearn` class (shown below) represents the information about individual words: the actual word, and what the student needs to learn about the word – whether they need to learn the spelling and/or the pronunciation of the word.

```
public class WordToLearn{
    private String word;
    private boolean spellingProblem;    // true if student must learn the spelling
    private boolean pronunciationProblem; // true if student must learn the pronunciation

    public WordToLearn(String wd, boolean sp, boolean pr){
        this.word = wd;
        this.spellingProblem = sp;
        this.pronunciationProblem = pr;
    }

    public String getWord (){
        return this.word;
    }

    public boolean isSProblem(){
        return spellingProblem;
    }
    public void learnS(){
        spellingProblem=false;
    }

    public boolean isPProblem(){
        return pronunciationProblem;
    }
    public void learnP(){
        pronunciationProblem=false;
    }
}
```

(Question 3 continued)

(a) [4 marks] The following test method tests the methods of the `WordToLearn` class. What will it print out to `System.out`?

```
public static void test(){
    WordToLearn wd1 = new WordToLearn("Straight", true, false);

    if (wd1.isSProblem()) {
        System.out.println("spell " + wd1.getWord() );
    }
    else {
        System.out.println("spelling done ");
    }

    WordToLearn wd2 = new WordToLearn("sleight", true, true);
    System.out.println("say " + wd2.getWord());
    wd2.learnP();
    if (wd2.isPProblem()) {
        System.out.println("say " + wd2.getWord() + " again");
    }
    else{
        System.out.println("spell " + wd2.getWord() );
    }
}
```

```
spell Straight
say sleight
spell sleight
```

(b) [4 marks] The `ESLTeacher` class uses the `WordToLearn` class, and contains an array to store the information about a collection of `WordToLeans`. Declare and initialise fields of the `ESLTeacher` class so that a `ESLTeacher` object could hold information on up to 200 `WordToLeans`. It should use an array and a count.

```
public class ESLTeacher{
    private final int maxWords = 200;
    private WordToLearn[] words = new WordToLearn[maxWords];
    private int count = 0;
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [6 marks] Complete the following listSpellingWords method in the ESLTeacher class so that it prints (to System.out) all the words that are spelling problems.

```
public void listSpellingWords(){  
    for (int i=0; i<this.count; i++){  
        if ( this.words[i].isSProblem() ) {  
            System.out.println(this.words[i].getWord());  
        }  
    }  
}
```

(d) [8 marks] Complete the following addSpellingWord method in the ESLTeacher class which should allow a user to add a new spelling word to the collection. If the array of words is full, the method should print The list is full and not add a word. Otherwise, the method should ask the user for the word (using System.in or JOptionPane), and add it to the list, marked as a spelling problem (but not a pronunciation problem).

```
public void addSpellingWord(){  
    if (this.count >= this.maxWords) { // or >= this.words.length  
        System.out.println("The list is full");  
    }  
    else {  
        String wd = JOptionPane.showInputDialog("What is the word??");  
        this.words[this.count] = new WordToLearn(wd, true, false);  
        this.count++;  
    }  
}
```

(Question 3 continued)

(e) [8 marks] Complete the following `learnPronunciation` method in the `ESLTeacher` class so that it allows the user to record that the user has learned to pronounce a word. It should find the given word in the list, and record that it is no longer a pronunciation problem. If the word is not a spelling problem, then the word should be removed from the list. If the given word is not in the list, it should print a message to say that the word was not found.

Note that the words do not need to be kept in any particular order, but there should never be null values in the first `count` elements of the array.

```
public void learnPronunciation(String wd){
    for (int i=0; i<this.count; i++){
        if ( this.words[i].getWord().equals(wd) ){
            this.words[i].learnP ();
            if ( ! this.words[i].isSPProblem() ){
                this.words[i] = this.words[this.count-1];
                this.count--;
            }
            return;
        }
    }
    System.out.println("Word "+wd+" not found");
}
```

Question 4. 2D Arrays

[30 marks]

This question concerns a program that helps a farmer plan out the amount of fertiliser to be applied to each of a collection of paddocks in each week over an 8 week period. The program stores the planned amounts in a 2D array, indexed by the paddock number (0 to `maxPaddocks-1`) and the week (0 to `maxWeeks`). The following table shows an example plan:

| Weeks | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|------|------|------|------|------|------|------|------|
| Paddock 0: | 94.6 | 84.6 | 13.1 | 77.5 | 57.6 | 45.4 | 38.0 | 21.5 |
| Paddock 1: | 94.6 | 84.6 | 77.5 | 63.1 | 57.6 | 45.4 | 38.0 | 21.5 |
| Paddock 2: | 94.6 | 77.5 | 77.5 | 63.1 | 63.1 | 45.4 | 38.0 | 21.5 |
| Paddock 3: | 18.4 | 50.1 | 28.2 | 46.2 | 31.3 | 98.5 | 8.0 | 83.5 |
| Paddock 4: | 72.8 | 64.4 | 57.4 | 43.2 | 31.5 | 21.8 | 14.8 | 6.1 |
| Paddock 5: | 20.4 | 51.1 | 74.0 | 17.6 | 69.5 | 50.0 | 5.0 | 8.2 |

The program has methods that allow the farmer to analyse and modify the current plan in a number of ways.

Field declarations and one method of the class are shown below:

```
public class FertiliserPlan {  
  
    private final int maxPaddocks = 10;  
    private final int maxWeeks = 8;  
    private double[][] plan = new double[maxPaddocks][maxWeeks];  
  
    public void printPlan(){  
        for (int p=0; p<maxPaddocks; p++){  
            System.out.printf("Paddock %d: ", p);  
            for (int w=0; w<maxWeeks; w++){  
                System.out.printf(" %5.1f ", this.plan[p][w] );  
            }  
            System.out.println ();  
        }  
    }  
}
```

(a) [6 marks] Complete the following multiply method so that it increases every amount in the current plan by the given factor. For example `multiply(2.5)` should multiply every value in the plan field by 2.5.

```
public void multiply(double factor){  
    for (int p=0; p<maxPaddocks; p++){  
        for (int w=0; w<maxWeeks; w++){  
            this.plan[p][w] = this.plan[p][w] * factor;  
        }  
    }  
}
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [6 marks] Complete the following `weeklySummary` method that will print out the total fertiliser required in each week of the plan, in the following format:

```
Week 0: 593.1 kg
Week 1: 618.4 kg
:
```

```
public void weeklySummary(){
    for (int w=0; w<maxWeeks; w++){
        double tot= 0;
        for (int p=0; p<maxPaddocks; p++){
            tot = tot+ this.plan[p][w];
        }
        System.out.printf("Week %d: %5.1f kg\n", w, tot);
    }
}
```

(c) [6 marks] A farm magazine recommends that the amount of fertiliser should always be lower than what was applied the previous week. Complete the following `checkDecreasing` method that searches the plan for paddocks in which the amount of fertiliser is NOT decreasing every week, and prints out a list of each such paddocks (identified by their numbers). For the example above, it should print 0, 2, 3, 5,

```
public void checkDecreasing(){
    for (int p=0; p<maxPaddocks; p++){
        for (int w=1; w<maxWeeks; w++){
            if (this.plan[p][w] >= this.plan[p][w-1]){
                System.out.println(p + " , ");
                break;
            }
        }
    }
}
```

(Question 4 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 4 continued)

(d) [6 marks] The farmer then decides the magazine is completely wrong and wants to be able to reverse the order of a plan. The following `reverse` method is intended to reverse the order of the amounts in the plan — for each paddock, it should swap the amounts in the first and last weeks, the second and second to last weeks, etc. For example, if the values in the plan for the first paddock (from week 0 to week 7) were:

5.0 9.0 1.0 3.0 4.0 7.0 2.0 8.0

then, after the `reverseOrder` method was called, the values ought to be:

8.0 2.0 7.0 4.0 3.0 1.0 9.0 5.0

```
public void reverseOrder(){    // NOTE: THIS VERSION HAS ERRORS!!!
    for (int w=1; w<maxWeeks; w++){
        for (int p=0; p<maxPaddocks; p++){
            this.plan[p][w] = this.plan[p][maxWeeks - w];
            this.plan[p][maxWeeks - w] = this.plan[p][w];
        }
    }
}
```

If the values in the plan for the first paddock were:

5.0 9.0 1.0 3.0 4.0 7.0 2.0 8.0

What would be the values in the plan really be after the `reverseOrder` method was called?

5.0 8.0 2.0 7.0 4.0 7.0 2.0 8.0

(e) [6 marks] Write a correct version of `reverseOrder`

```
public void reverseOrder(){
    for (int w=0; w<maxWeeks/2; w++){
        for (int p=0; p<maxPaddocks; p++){
            double temp = this.plan[p][maxWeeks - w - 1];
            this.plan[p][maxWeeks - w - 1] = this.plan[p][w];
            this.plan[p][w] = temp;
        }
    }
}
```

Question 5. Designing with Interface Classes

[30 marks]

Suppose you are implementing a `GuestHouse` program for a guest house that accepts bookings from individuals and from travel agents. You have already implemented the following two classes for representing information about the two kinds of bookings.

```
public class IndividualBooking{
    private String name;
    private String phone;
    private int guests;

    public IndividualBooking(String name, int guests, String phone){
        this.name = name;
        this.guests = guests;
        this.phone = phone;
    }

    public String getName(){ return this.name; }
    public int getNumGuests(){ return this.guests; }
    public void print(){
        System.out.printf("%s: %d guests, ph %s", this.name, this.guests, this.phone);
    }
    // plus further methods
}

public class AgentBooking{

    private String name;
    private int guests;
    private String agent;

    public AgentBooking(String name, int guests, String agent){
        this.name = name;
        this.guests = guests;
        this.agent = agent;
    }

    public String getName(){ return this.name; }
    public int getNumGuests(){ return this.guests; }
    public void print(){
        System.out.printf("%s: %d guests, (agent %s)", this.name, this.guests, this.agent);
    }
    // plus further methods
}
```

(Question 5 continued on next page)

(Question 5 continued)

(a) [4 marks] Suppose you have written the following field declaration and test method for the GuestHouse program. The code does not compile. Specify the part(s) of the code that the compiler would complain about, and explain why it would complain.

```
private Object[ ] bookings = new Object[20];
//other fields and methods ....

public void test(){
    this.bookings[0] = new IndividualBooking("Jones", 2, "1239876");
    this.bookings[1] = new AgentBooking("Smith", 5, "TravelSites");
    for (int i = 0; i < 2; i++){
        if (this.bookings[i].getNumGuests() > 1) {
            this.bookings[i].print();
        }
    }
}
```

The compiler will complain about `bookings[i].getNumGuests()` and `bookings[i].print()` because Objects do not have the methods `getNumGuests()` and `print()`

(b) [5 marks] If you wanted to use an interface class to enable the GuestHouse program to store a collection of IndividualBooking and GroupBooking objects, explain what would be in the interface class, and what you would have to change in the other classes and the test method to make it work.

The interface class (eg, called **Booking**) must contain the headers of the three methods `getNumGuests`, `getName`, and `print`.
 Declare in their class header that `AgentBooking` and `IndividualBooking` implement the **Booking** interface,
 Change the declaration of the `bookings` array to
`Booking[] bookings = new Booking[20];`
 You don't have to change the test method.

(Question 5 continued on next page)

(Question 5 continued)

The rest of the question concerns a program for a maze game with several kinds of characters that can move around the maze: Players, AiBots, and Elephants.

- Player characters are controlled by the users;
- AiBots control themselves according to a strategy, and are each owned by a particular player character,
- Elephants aren't owned by anything, and stomp around the maze randomly.

As a result of collisions, characters may lose some of their energy, and will eventually be eliminated from the game. The winner is the player who survives the longest.

Each place in the maze is identified by a number (an integer).

All Characters (Players, AiBots, and Elephants) have methods to:

- move: move one square in their current direction;
- turn: to the left or right
- collide with another character: producing interesting visual effects, and reducing their energy levels.
- return their current energy level (a double)
- return their current position in the maze (an integer)

(c) [5 marks] Define an appropriate interface class `Character`, specifying all the method headers.

```
public interface Character{
    public void move();
    public void turn(String direction );
    public void collide (Character ch);
    public double energyLevel();
    public int position ();
}
```

(Question 5 continued on next page)

(Question 5 continued)

(d) [4 marks] An AiBot character needs to store

- its position,
- its current energy level,
- its current direction (which could be north, south, east, or west),
- its strategy (which could be aggressive, fearful, or tricky), and
- its owner.

Complete the following header of the AiBot class and the declarations of fields to store this information. (You do not need to initialise the fields.)

```
public class AiBot implements Character{  
    private int position;  
    private double energy;  
    private String direction;  
    private String strategy;  
    private Player owner;
```

(Question 5 continued on next page)

(Question 5 continued on next page)

(Question 5 continued)

(e) [12 marks] The `MazeGame` class stores the collection of `Characters` in an array. The relevant fields in the `MazeGame` class are:

```
private final int maxChars = 30;  
private Character[] characters = new Character[maxChars];
```

Part of what must happen in each step of the game is that the game must make each character move a step. It must then check whether any characters are in the same maze position as any other character, and if so, make them collide with each other. Note that if more than two characters are at the same position, they should each collide with all the other characters at the position. Finally, it should remove any `Characters` whose energy is below 0.

Complete the `moveStep` method of the `MazeGame` class on the facing page to do this.

(Question 5 continued on next page)

(Question 5 continued)

```
public void moveStep(){
    for (int i = 0; i < this.characters.length; i++){
        if (this.characters[i] != null) {
            this.characters[i].move();
        }
    }
    for (int j = 0; j < this.characters.length; j++){
        Character ch = this.characters[j];
        if (ch != null) {
            for (int k = j+1; k < this.characters.length; k++){
                Character other = this.characters[k];
                if (other != null && ch.position() == other.position()){
                    ch.collide(other);
                    other.collide(ch);
                }
            }
        }
    }
    for (int i=0; i<this.characters.length; i++){
        if (this.characters[i] != null && this.characters[i].energyLevel() <= 0){
            this.characters[i] = null;
        }
    }
}
```

Question 6. Recursion

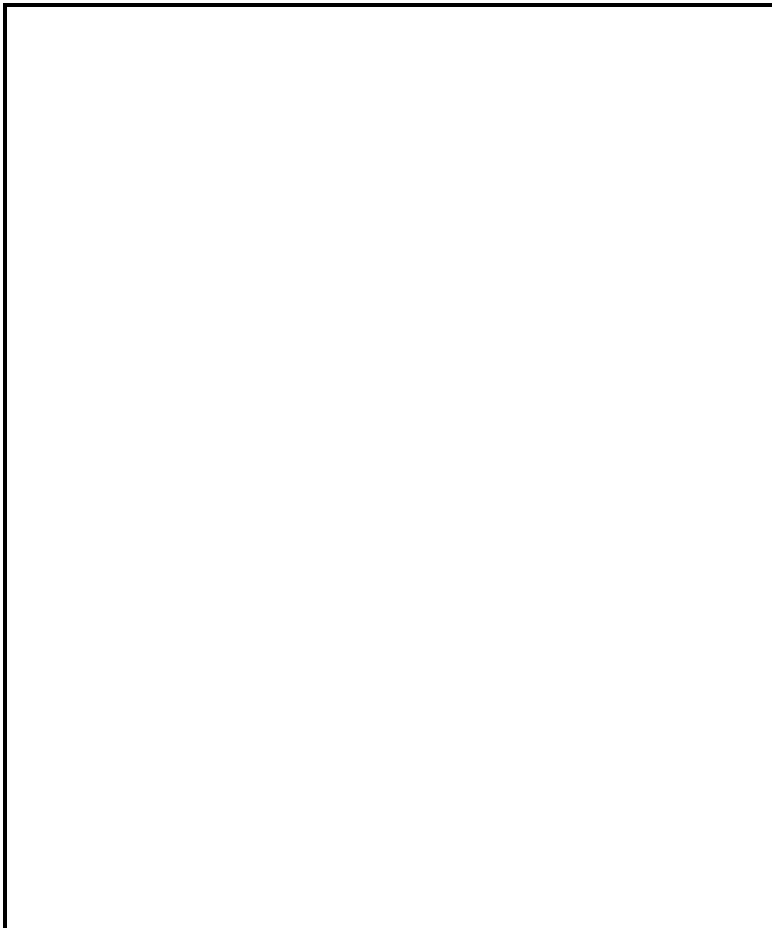
[7 marks]

The following recursive method, `recDraw`, draws rectangles on a `DrawingCanvas`. In the boxes below, sketch what it would draw on the `DrawingCanvas` and what it would print in the terminal window if it were called by the statement

```
this.recDraw(80, 100);
```

```
public void recDraw(int size, int off){
    System.out.println("draw: "+ size +"@" + off)
    this.canvas.drawRect(off, off, size, size);
    if (size > 20){
        this.recDraw(size/2, off);
        this.recDraw(size/2, off+size);
    }
}
```

Assume that the `DrawingCanvas` is on the left and the terminal window is on the right:



```
draw: 80 @ 100 draw:
40 @ 100 draw: 20 @
100 draw: 20 @ 140
draw: 40 @ 180 draw:
20 @ 180 draw: 20 @
220
```
