



EXAMINATIONS — 2007

MID-YEAR

COMP 102

WITH SOLUTIONS

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.

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

Questions

	Marks
1. Understanding Java	[56]
2. Programming with Loops	[15]
3. Files	[17]
4. Implementing Collections	[15]
5. Using Collections	[12]
6. Designing with Interfaces	[32]
7. GUIs and Mouse Events	[12]
8. Recursion	[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

[56 marks]

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

```

int x = 5;
int y = 3;
y = x + 4;
x = y / 2;
System.out.printf("x= %d y= %d expr= %4.2f\n", x, y, y/2.0);

```

```

| x= 4 y= 9 expr= 4.50

```

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

```

public int compute(int a, int b) {
    if (a < 8 || b == 5)
        return a;
    if (a > 5 && b <= 7)
        return b;
    if (a == b * 2)
        return a + b;
    return 0;
}

```

What would the following calls to compute return?

```

compute(6, 3) ==> | 6
compute(20, 10) ==> | 30
compute(10, 15) ==> | 0
compute(10, 5) ==> | 10
compute(8, 6) ==> | 6

```

(Question 1 continued on next page)

(Question 1 continued)

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

```
int n = 2;
while ( n < 16 ){
    System.out.printf("n = %d\n", n);
    if ( n > 8)
        break;
    n = n + 2;
}
System.out.printf("Finally, n = %d\n", n);
```

```
n = 2
n = 4
n = 6
n = 8
n = 10
Finally, n = 10
```

(d) [5 marks] Suppose the variable `words` is declared to be an array containing 10 strings.

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

words:

bee	cow	ant	car	cat	bat	ape	can	bin	gnu
0	1	2	3	4	5	6	7	8	9

What will the following code fragment print out?

```
for ( int index=0; index < words.length; index++ ){
    if ( words[index].startsWith("c") ){
        System.out.println(words[index + 1]);
    }
}
```

```
ant
cat
bat
bin
```

(Question 1 continued on next page)

(Question 1 continued)

(e) [5 marks] Show how you could reorder the Strings in the `words` array of the previous question so that the Java fragment above would result in an error.

Words:

bee	cow	ant	car	cat	bat	ape	gnu	bin	can
0	1	2	3	4	5	6	7	8	9

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

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

Show the contents of `words` after the following `remove` method is called with the arguments `remove(words, 4)`.

```
public void remove(String [ ] words, int index){
    for ( int i=words.length-2; i>index; i-- ){
        words[i] = words[i+1];
    }
}
```

words:

bee	cow	ant	car	cat	gnu	gnu	gnu	gnu	gnu
0	1	2	3	4	5	6	7	8	9

(Question 1 continued on next page)

(Question 1 continued)

The Elf class on the facing page defines Elf objects, which have two fields to store their age and hat colour. The class also contains a test method.

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

```
A: Blue elf (3)
B: Blue elf (10)
C: Green elf (5)
D: Green elf (5)
E: Yellow elf (11)
```

(h) [5 marks] Write an additional method called `grownUp` for the Elf class which returns `true` if the elf is 11 or older, and `false` if the elf is under 11.

```
public boolean grownUp(){
    return (this.age >= 11);
}
```

(Question 1 continued on next page)

(Question 1 continued)

```
public class Elf{
    private int age;
    private String hatColour;

    public Elf( int a){
        this.age = a;
        this.hatColour = "Blue";
    }

    public void changeHat(String c){
        this.age++;
        this.hatColour = c;
    }

    public String toString(){
        return (this.hatColour + " elf (" + this.age + ") ");
    }

    public static void test(){
        Elf jim = new Elf(3);
        Elf jak  = new Elf(10);

        System.out.println("A: " + jim.toString());
        System.out.println("B: " + jak.toString());

        jim.changeHat("Red");
        jim.changeHat("Green");
        System.out.println("C: " + jim.toString());

        jak.changeHat("Yellow");
        System.out.println("D: " + jim.toString());
        System.out.println("E: " + jak.toString());
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

(i) [6 marks] Suppose the file "command.log" contains the following text:

```
ls
makeDemo 10 MiniDraw
copy assig
cpRec Initial/doc assignment-code/lab-10/MiniDraw
inkscape-drawingFile
xlbiff
```

What will the following printFile method print out if it is called by printFile("command.log")?

```
public void printFile (String fname){
    try{
        Scanner scan = new Scanner (new File(fname));
        int a = 0;
        while ( scan.hasNext() ){
            String str = scan.nextLine();
            if ( str.length() < 12 )
                System.out.printf ("%d: %s\n", a, str);
            a++;
        }
        System.out.println(scan.next ());
        scan.close();
    }
    catch(Exception e){System.out.println("File reading failed");}
}
```

```
0:  ls
2:  copy assig
5:  xlbiff
File reading failed.
```

(Question 1 continued on next page)

(Question 1 continued)

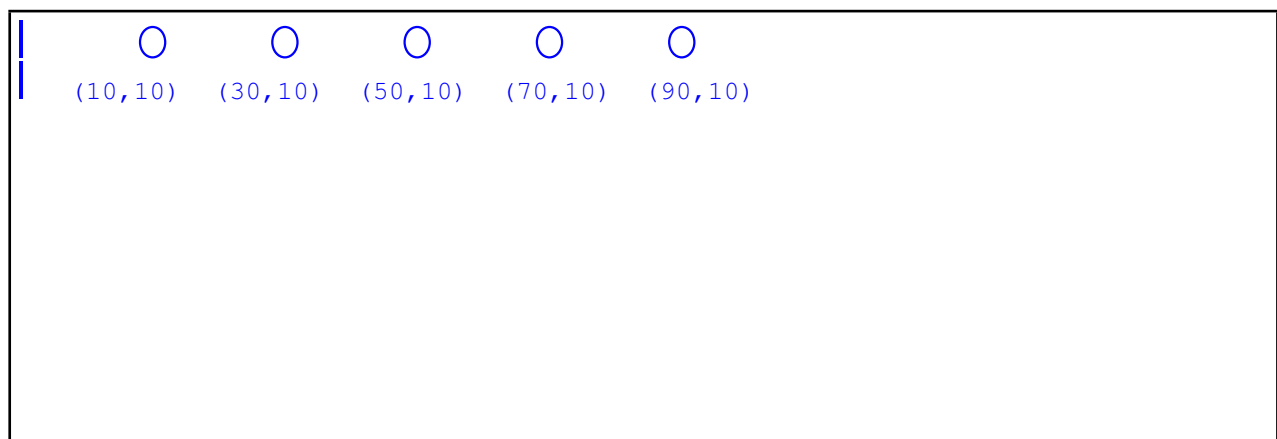
(j) [5 marks] Complete the following `countNegatives` method that is given an array of doubles, and will **return** the number of cells in the array that contain negative numbers.

```
public int countNegatives(double[] data){
    int count = 0;
    for (int i=0; i<data.length; i++){
        if ( data[i] < 0 )
            count++;
    }

    return count;
}
```

(k) [5 marks] Sketch what the following `drawStuff` method would draw on the canvas if it were called with `drawStuff(10, 10, 4)`. Assume that `canvas` is a field containing a `DrawingCanvas` object. Write the top-left coordinates of each Oval on your sketch.

```
public void drawStuff(int x, int y, int num){
    this.canvas.drawOval(x, y, 8, 12);
    if (num > 0){
        this.drawStuff(x+20, y, num-1);
    }
}
```



Question 2. Programming with Loops

[15 marks]

The following `printTable(n)` method is intended to print out a multiplication table up to n , showing each answer just once. For example, if the argument is 4, it should print out

```
1: 1  2  3  4
2:   4  6  8
3:    9 12
4:   16
```

If the argument is 5, it should print out

```
1: 1  2  3  4  5
2:   4  6  8 10
3:    9 12 15
4:   16 20
5:   25
```

```
/* This version of printTable has errors */
public void printTable( int n) {
    for( int row= 0; row<n; row++){
        System.out.printf( "%d: ", row);
        for( int col = row; col<n; col++){
            System.out.printf( "%2d ", row*col);
            if (col > n)
                System.out.println ();
        }
    }
}
```

(a) [5 marks] The version of `printTable` above has errors. What does it print out if it is called with an argument of 4?

```
| 0:  0 0 0 0 1:  1 2 3 2:  4 6 3:  9
```

(Question 2 continued on next page)

(Question 2 continued)

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

```
public void printTable (int n){
    for (int row= 1; row<=n; row++){
        System.out.printf ("%d: ", row);
        for (int col = 1; col <=n; col++){
            if (col>= row)
                System.out.printf ("%2d ", row*col);
            else
                System.out.print("  ");
        }
        System.out.println ();
    }
}
```

```
}
```

Question 3. Files

[17 marks]

Suppose the file "DutyHours.txt" contains data about hours of duty where each line contains a day of the week, a date, and pairs of times giving the start and end of each duty session. For example, the file might contain

```
Mon 07/05/2007 1000 1200 1700 1800
Wed 09/05/2007 1400 1700
Fri 11/05/2007
Mon 14/05/2007 0900 1100 1600 1900 2100 2200
```

The first line says that there are two duty sessions (from 10am to 12 noon and from 5pm to 6pm) on Monday the 7th May 2007.

The start and end times are given in 24-hour time and duties always start and end on the hour.

The number of duties on each day varies and there are days that have no duties.

(a) [5 marks] What will the following method print to System.out?

```
public void testFiles () {
    try{
        Scanner fileScan = new Scanner(new File("DutyHours.txt"));
        String day = fileScan.next ();
        String date = fileScan.next ();
        while (fileScan.hasNextInt()){
            System.out.println (fileScan. nextInt ());
        }
        fileScan.close ();
    }
    catch(IOException e){System.out.println("File reading failed: "+e);}
}
```

```
1000
1200
1700
1800
```

(Question 3 continued on next page)

(Question 3 continued)

(b) [12 marks] Write a method that reads data from a file in the format described above and calculates the total working hours (duty hours). On the example file above, it should return 12.

```
public int workingHours() {
    try{
        Scanner fileScan = new Scanner(new File("DutyHours.txt"));
        int hours = 0;
        while(fileScan.hasNext()) {
            String day = fileScan.next();
            String date = fileScan.next();
            while (fileScan.hasNextInt()){
                int start = fileScan.nextInt();
                int end = fileScan.nextInt();
                hours = hours + (end-start)/100;
            }
        }
        fileScan.close();
        return hours;
    }
    catch(IOException e){System.out.println("File reading failed: "+e);}
}
```

Question 4. Implementing Collections

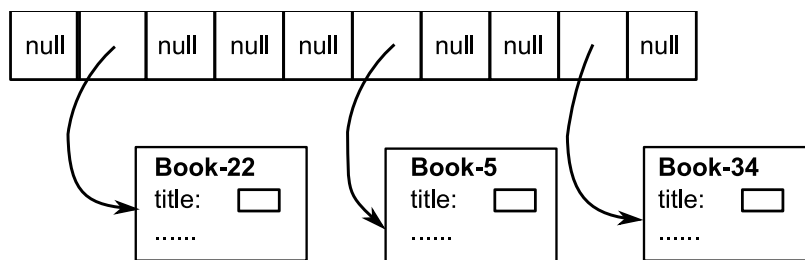
[15 marks]

This question (and the next) concern a `BookTracker` program to help a user keep track of their rare book collection. One class in the program is the `BookCollection` class, which uses an array to store the information about the current collection of books. It has the following two fields:

```
private static final int MaxBooks = 1000;  
private Book[] bookArray = new Book[MaxBooks];
```

Note that `Book` is a class for representing information about an individual book.

Initially, every cell of the array will contain `null`. When you add a book to the collection, you can put it in any cell that currently contains `null`; when you remove a book, you simply replace it by `null`. For example, if the array had a length of 10, and contained three books, it might look like the following:



The `BookCollection` class has several methods, including:

- `size()`, which returns the number of books in the collection.
- `addBook(Book b)`, which adds a new book to the collection.
- `findBook(String str)`, which returns the book in the collection whose title matches the string.
- `printAll(PrintStream ps)`, which prints information about each book to a `PrintStream` (eg, `System.out`).

(a) [5 marks] Complete the `size()` method of the `BookCollection` class so that it returns the number of books in the collection.

```
public int size(){  
    int count=0;  
    for( int i = 0; i < bookArray.length; i++) {  
        if (bookArray[i]!=null )  
            count++;  
    }  
    return count;  
}
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [5 marks] Complete the following `addBook()` method of the `BookCollection` class so that it adds the given book to the collection. If the collection is full, it should do nothing.

```
public void addBook(Book b){  
    for( int i= 0; i < bookArray.length; i++) {  
        if (bookArray[i]==null){  
            bookArray[i]= b;  
        }  
    }  
}
```

(c) [5 marks] Complete the `findBook` method below so that it returns the book in the collection that has the given title. If there is no such book, it should return `null`. You may assume that all the books have different titles.

Assume that the `Book` class includes a `hasTitle` method that returns `true` if a book has a title that matches a given string, and `false` otherwise:

```
public boolean hasTitle(String str) .....
```

```
public Book findBook(String title ){  
    for( int i= 0; i < bookArray.length; i++) {  
        if ( bookArray[i]!=null && bookArray[i].hasTitle( title ) )  
            return bookArray[i];  
    }  
    return null;  
}
```

Question 5. Using Collections

[12 marks]

The `BookTracker` program of the previous question records information about a rare book collection. It allows a user to record when a book is added to the collection, borrowed temporarily, or sold from the collection. The program has three classes:

- `BookTracker`, which handles all interaction with the user,
- `BookCollection`, (from the previous question) for storing the collection of books,
- `Book`, which represents information about an individual book.

All the relevant method headers for the `Book` and `BookCollection` classes are given on the facing page. In your answers below, you should use appropriate methods from the `Book` and/or `BookCollection` classes.

(a) [4 marks] Assume that the `BookTracker` class contains a field:

```
public BookCollection books = new BookCollection();
```

Complete the following `display` method in the `BookTracker` class so that it prints out the number of books and descriptions of each book in the collection to `System.out`.

```
public void display(){  
    System.out.println(this.books.size() + " Books:");  
    this.books.printAll (System.out);  
  
}
```

(b) [8 marks] Complete the following `checkStatus` method of the `BookTracker` class so that it prompts the user for the title of a book (using `System.out` and `System.in`), and then prints out

- "Unknown" if the book is not in the collection,
- "Available" if the book is in the collection and is not currently on loan, or
- "On Loan" if the book is in the collection but is currently borrowed by someone.

```
public void checkStatus(){  
    System.out.print("Title: ");  
    Scanner scan = new Scanner(System.in);  
    String title = scan.nextLine();  
    Book b = books.findBook(title);  
    if (b==null)  
        System.out.println("Unknown");  
    else if (b.onLoan())  
        System.out.println("On Loan");  
    else  
        System.out.println("Available");  
  
}
```

(Question 5 continued)

```

public class Book {

    private String title ;      ....

    /** Prints a description of the book to a PrintStream (eg System.out) */
    public void print (PrintStream ps){....

    /** Returns true if the book has the specified title , false otherwise */
    public boolean hasTitle(String str) {....

    /** Returns true if the book is on loan to someone */
    public boolean onLoan() {...

        :

    }

```

```

public class BookCollection{

    /** Returns the number of books in the collection */
    public int size(){ ....

    /** Adds a book to the collection */
    public void addBook(Book b){ ....

    /** Returns the book in the collection that has the specified title ,
        or null if there is no such book */
    public Book findBook(String title){ ....

    /** Removes from the collection the book that has the specified title */
    public void removeBook(String title){ ....

    /** Prints descriptions of each book to a PrintStream (eg System.out) */
    public void printAll (PrintStream ps){
        ....
    }

```

Question 6. Designing with Interfaces

[32 marks]

(a) [2 marks] An interface class specifies a type. What kinds of things will be declared/defined in an interface class? (Note, this does not refer to user interfaces).

```
| Method headers
```

(b) [3 marks] An ordinary class specifies more than an interface class. List three kinds of things that can be declared/defined in an ordinary class, but cannot be present in an interface class.

```
| Method bodies  
| Field definitions  
| Constructors
```

The rest of this question concerns part of a program for a computer game in which players wander around rooms, picking up items that they find in the rooms and putting them in their “carry-bag”. There are several different categories of items, including tools, food, and maps. You have chosen to define three classes (Tool, Food, and Map) for these different categories. The different categories have different fields for storing information and different methods for “using” the items. However, they all have the following three methods.

- `print(PrintStream ps)`, for printing a description of the item to a `PrintStream` (such as `System.out`)
- `getWeight()`, which returns the weight of the item (a double),
- `makeWet()`, which modifies the state of the item as a result of making it wet (for example, food may become spoiled, maps may become soggy)

You have also chosen to define a `CarryBag` class to represent the collection of items in a player’s carry-bag.

(c) [5 marks] Define an interface class to represent all categories of items, specifying the three methods that can be called on them all.

```
public interface Item{  
  
    public void print(PrintStream ps);  
    public double getWeight();  
    public void makeWet();  
  
}
```

(d) [4 marks] The CarryBag class has a field `items` containing an array for the items, along with a field for the count of the items. Give appropriate declarations for these fields so that the array could hold Tool, Food, and/or Map items. The field declarations should initialise the fields appropriately.

```
public class CarryBag{
    private int MaxItems = 100;
    private Item[] items = new Item[MaxItems];
    private int count;
```

The Map class represents the map items that the player finds. A map will have a weight (usually small), a material that it is printed on (*e.g.*, paper or leather), a size (length and width in centimeters) and a locality (the name of the area shown in the map). It will also have a description of its current state, such as “badly torn”, or “soggy and illegible”. It uses an enum to represent the different kinds of material.

(e) [7 marks] Complete the header and the field declarations of the Map class to represent this information.

```
public class Map implements Item {

    public enum Material {Paper, Leather, Vellum, Copper, Wood};
    private int length;
    private int width;
    private double weight;
    private String locality ;
    private Material material;
    private String currentState;
```

(f) [4 marks] Write the `getWeight()` method of the Map class so that it returns the weight of a map.

```
public double getWeight(){
    return weight;
```

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)

(g) [7 marks] Write the print method of the Map class so that it will print out an informative description of a map.

```
public void print(PrintStream ps){  
    ps.printf("%d x %d %s map of %s, weighing %2.1fgm. Condition: %s",  
        length, width, material, locality , weight, currentState);  
}
```

Question 7. GUIs and Mouse Events.

[12 marks]

Suppose you are writing a program that allows the user to draw polygons on the window by clicking the mouse at each vertex of the polygon until the user clicks on the starting point again.

Write code to respond to the mouse and draw polygons in this way. For every click (except the first one of each polygon), it should draw a line from the previous vertex to the new vertex. After the user has clicked again on the first vertex, it should let the user start another polygon.

Assume that there is a field `canvas` containing a `DrawingCanvas`. You only need to define the `mouseReleased` method and any additional fields it needs.

Note, the code only has to draw the polygon on the screen; it does not have to store the polygon anywhere.

```
public class PolygonDrawer implements MouseListener{
    .....
    // Field(s) for Mouse handling
    private boolean inPoly = false;
    private int firstx ;
    private int firsty ;

    private int lastx ;
    private int lasty ;

    // Method for mouse handling

    public void mouseReleased(MouseEvent e){
        int x = e.getX();
        int y = e.getY();
        if (inPoly){
            this.canvas.drawLine(lastx, lasty, x, y);
            lastx = x;
            lasty = y;
            if (x==firstx && y == firsty){
                inPoly = false;
            }
        }
        else {
            firstx = x;
            firsty = y;
            lastx = x;
            lasty = y;
            inPoly = true;
        }
    }
}
```

Question 8. Recursion

[21 marks]

(a) [7 marks] The following recursive method, `countdown`, should count down from its argument to 1, print "Bang" and then count up again. For example, if called with an argument of 5, it should print:

5 4 3 2 1 Bang 1 2 3 4 5

If called with an argument of 0 or less, it should just print "Bang".

Write `countdown` using recursion; do not use a `while` or a `for` loop.

```
public void countdown(int n){
    if (n == 0)
        System.out.print("Bang ");
    else {
        System.out.print(n+ " ");
        countdown(n-1);
        System.out.print(n+ " ");
    }
}
```

(Question 8 continued on next page)

(Question 8 continued)

(b) [10 marks] What will be printed out if the fill method below is called with the arguments fill(3,6)?

```
public void fill (int row, int col){
    fillSquare (row, col);
    if (row < col){
        fill (row, col-2);
        fill (row+1, col);
    }
    else if (row > col){
        fill (row-1, col);
        fill (row, col+1);
    }
}

public void fillSquare (int row, int col){
    System.out.printf (" (%d, %d) \n", row, col);
}
```

```
(3, 6)
(3, 4)
(3, 2)
(2, 2)
(3, 3)
(4, 4)
(4, 6)
(4, 4)
(5, 6)
(5, 4)
(4, 4)
(5, 5)
(6, 6)
```

(Question 8 continued on next page)

(Question 8 continued)

(c) [4 marks] Is the fill method guaranteed to stop on all possible inputs, or might it keep going for ever on some inputs? Justify your answer.

It will always stop.

It will not call itself again if the arguments are equal. If col is smaller than row, then in both of the recursive calls col is equal to row or it is closer to row (but is still smaller than row) so the calls must eventually converge.

If col is larger than row, then in one of the recursive calls, the arguments are either closer or equal, and in the other recursive call, the arguments are closer or equal, unless $col = row + 1$, in which case, col is smaller than row in the new call so that they will converge.

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
```