

MID-YEAR

<p>COMP 102 INTRODUCTION TO COMPUTER PROGRAM DESIGN</p>
---

Time Allowed: 3 Hours

Instructions: Attempt ALL Questions.

The exam will be marked out of 180.

Non-programmable calculators without full alphabet keys are permitted.

Non-electronic foreign language dictionaries are permitted.

Answer in the appropriate boxes if possible. If you write your answer elsewhere, make it clear where your answer can be found.

There is documentation at the end of the paper.

## Questions

	<b>Marks</b>
1. Understanding simple Java programs	[28]
2. Simple graphical objects	[30]
3. Arrays and files	[30]
4. Arrays of objects	[20]
5. Understanding composition	[30]
6. Program design with containers	[30]
7. Recursion	[12]

## Documentation (at end of paper)

- Methods for input and output with *UI*.
- Methods for event driven input with *UI*.
- Operators.
- Reading from files.
- Methods on *Strings*.
- The *Math* class.
- The *Trace* class.

**Question 1. Understanding simple Java programs**

[28 marks]

In this question, `win` is assumed to contain a UI object.

(a) [4 marks] What will be printed when method `Rat` is called?

```
public void Rat() {
    int k = 0;
    while (k < 8) {
        win.print(k + " ");
        k = k+2;
    }
}
```

(b) [4 marks] What will be printed when method `Mole` is called?

```
public void Mole() {
    for (int k = 30; k>26; k--) {
        win.print(k + " ");
    }
}
```

(c) [4 marks] What will be printed when method `Toad` is called with 7 as its argument?

```
public void Toad(int n) {
    for (int k = 0; k<n; k++) {
        if ( k < n/2 )
            win.print((k+1) + " ");
        else
            win.print((n-k+1) + " ");
    }
}
```

**(d)** [8 marks] Consider the following method definition:

```
public void Badger(int n) {  
    if ( n < 2 )  
        if ( n > 0 )  
            win.print("Ferret");  
        else  
            win.print("Weasel");  
    else  
        if ( n < 8 && n > 4 )  
            if ( n%2 == 0 )  
                win.print("Stoat");  
            else  
                if ( n-6 < 0 )  
                    win.print("Mouse");  
                else  
                    win.print("Rabbit");  
    }  
}
```

**(i)** [2 marks] What will be printed when `Badger` is called with 1 as its argument?

**(ii)** [2 marks] What will be printed when `Badger` is called with 6 as its argument?

**(iii)** [2 marks] What argument value, if any, would cause `Badger` to print `Mouse`?

**(iv)** [2 marks] What argument value, if any, would cause `Badger` to print `Weasel`?

(e) [4 marks] Consider the following class definition:

```
class Counter1 {
    private UI win;
    public Counter1() {
        win = new UI("Counter1");
        Count();
        Count();
    }
    public void Count() {
        int k = 1;
        for (int i = 1; i<5; i++) {
            win.print(k + " ");
            k++;
        }
    }
}
```

What will be printed when the constructor Counter1 is called?

(f) [4 marks] Consider the following class definition, which is the same as Counter1 except for the declaration of k:

```
class Counter2 {
    private UI win;
    private int k = 1;
    public Counter2() {
        win = new UI("Counter2");
        Count();
        Count();
    }
    public void Count() {
        for (int i = 1; i<5; i++) {
            win.print(k + " ");
            k++;
        }
    }
}
```

What will be printed when the constructor Counter2 is called?

**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. Simple graphical objects

[30 marks]

Suppose you are writing a program that involves drawing patterns of coloured squares on the screen. All squares are 50 pixels wide, and positioned within a screen area 400 pixels wide and 400 pixels high.

You are given the following class definition, which is used to represent squares on the screen:

```
/**
 * Class Square, for representing squares that can be drawn on the screen.
 * Constructor Square(UI u, double x, double y, colour c) makes a square of
 * colour c, with top-left corner at screen position (x,y) on UI window u.
 */

public class Square {

    private UI win;
    private double width = 50;
    private double posx, posy;
    private Colour colour;

    Square(UI u, double x, double y, Colour c) {
        win = u; posx = x; posy = y; colour = c;
        show();
    }

    /** Set square colour to c. */
    public void setColour(Colour c) {
        colour = c;
    }

    /** Move square down by dx and right by dy. */
    public void move(double dx, double dy) {
        erase();
        posx = posx + dx; posy = posy + dy;
        show();
    }

    /** Test whether this square touches, or overlaps with, another square */
    public boolean touching(Square that) {

        // Not implemented yet (see part (e))

    }

    /** Display the square. */
    private void show() {
        win.fillRect(posx, posy, width, width, colour);
    }

    /** Erase the square. */
    private void erase() {
        win.eraseRect(posx, posy, width, width);
    }

}
```

**Write Java code to perform the following actions, using the `Square` methods where appropriate.**

**(a)** [4 marks] Declare two `Square` variables, and initialise them so as to draw a black square in the top-left hand corner of the screen and a grey square immediately to its right, as shown in Figure 1.

(Note that the colours black and grey are specified as `Colour.black` and `Colour.grey`, respectively.)

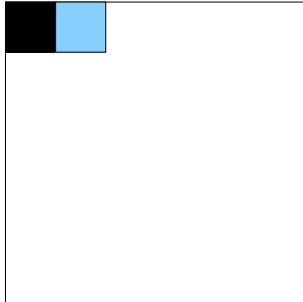


Figure 1

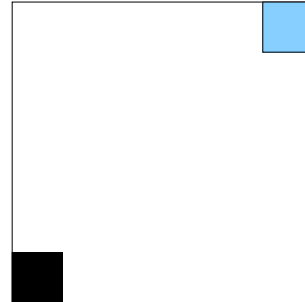
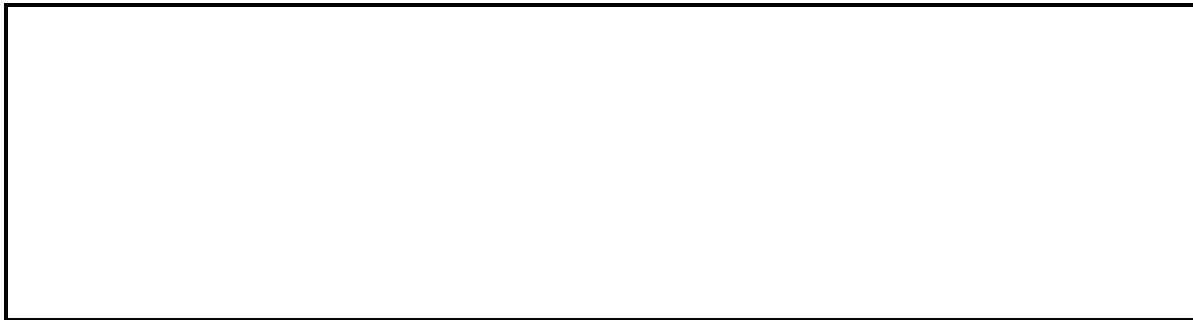
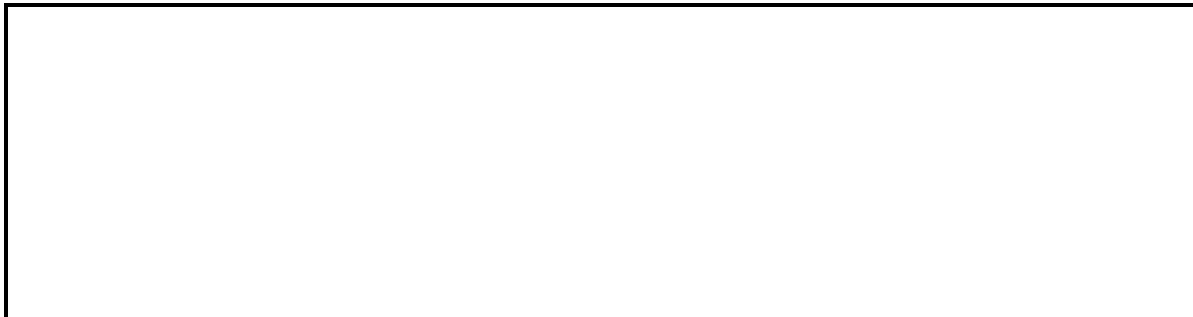


Figure 2



**(b)** [4 marks] Move the black square to the bottom-left hand corner of the screen, and the grey square to the top-right hand corner of the screen, as shown in Figure 2.



(c) [8 marks] Move the two squares horizontally along the top and bottom of the screen until the black square is in the bottom-right hand corner and the grey square is in the top-left hand corner, as shown in Figure 3. The squares should be moved in a series of steps of 10 pixels each (as indicated by the dashed lines in Figure 3).

You should use a for loop to move the squares, working out beforehand how many steps will be required to move the squares to their new positions.

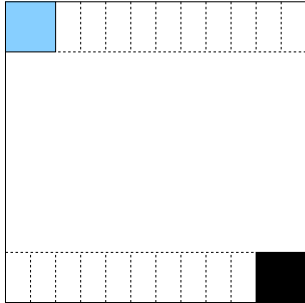


Figure 3

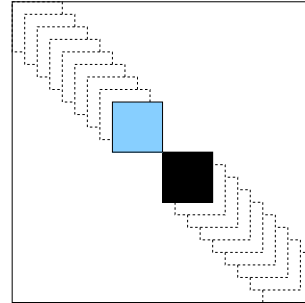
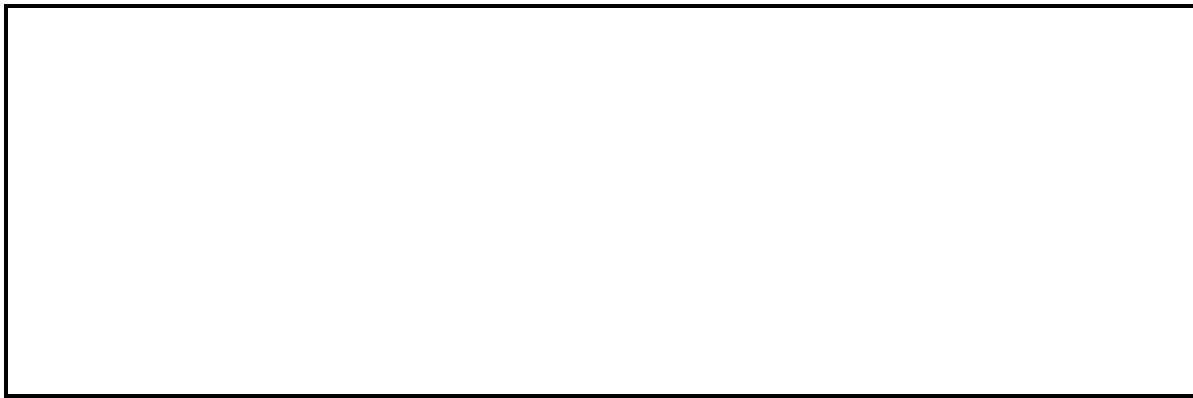


Figure 4



(d) [8 marks] Move the two squares diagonally towards each other, until their corners touch, as shown in Figure 4. The squares should be moved in a series of steps, each moving the square 10 pixels in both the x and y direction (as indicated by the dashed lines in Figure 4).

You should use a while loop to move the squares, calling the method `touching` to determine when to exit from the loop (see part (e)).



(e) [6 marks] Complete the definition of the touching method in the Square class, so that a call `s1.touching(s2)` will return `true` if squares `s1` and `s2` are touching (or overlapping), and `false` otherwise.

```
public boolean touching(Square that) {
```

```
}
```

### Question 3. Arrays and files

[30 marks]

(a) [20 marks] Array of numbers

We create an array with 5 numbers and declare it as a data field in a class.

```
int[] numbers = {40, 30, 20, 50, 80};
```

The array looks like this:

numbers[0]	numbers[1]	numbers[2]	numbers[3]	numbers[4]
40	30	20	50	80

Note: win is a UI object and it is declared as a data field in the class.

(i) [5 marks] What will the following method print out?

```
public void print1(){  
    int num = numbers[0] + numbers[1];  
    win.println(num);  
    int i = 1;  
    win.println(numbers[2 * i + 1]);  
}
```

(ii) [5 marks] What will the following method print out?

```
public void print2(){  
    for (int i = 0; i < 5; i++){  
        if (numbers[i] <= 50){  
            win.println(numbers[i]);  
        }  
    }  
}
```



**(b)** [10 marks] Reading from files

Suppose we have a file containing data about students. The data for each student is stored in one line, giving login name, ID number and the papers (courses), followed by "END", to signal the end of papers for that student. For example, the following file contains data for five students.

```
pjohnston 1001 comp130 comp102 comp103 info101 END
jclinton 3002 comp302 comp307 END
srobson 1003 comp102 END
dsmith 1004 comp102 comp103 comp201 END
kwilliams 3005 comp304 comp305 comp307 END
```

**Note:** You must answer the questions using the example file given above. Assume win is a UI object.

**(i)** [5 marks] What will the following count method print out?

```
public void count(){
    DataFileReader file = new DataFileReader();
    int count = 0;
    while ( !file.endOfFile() ) {
        String s = file.readToken();
        if (s.equals("comp102")){
            count++;
        }
    }
    win.println(count);
    file.close();
}
```

(ii) [5 marks] What will the following show method print out?

```
public void show(){
    DataFileReader file = new DataFileReader();
    while (!file.endOfFile()){
        String s1 = file.readToken();
        String s2 = file.readToken();
        String p = file.readToken();
        int count = 0;

        while (!p.equals("END")){
            count++;
            p = file.readToken();
        }
        win.println(s2 + ": " + count);
    }
    file.close();
}
```

#### Question 4. Arrays of objects

[20 marks]

This question involves the program `HotelProgram` which is used by hotel managers to record data about customers. In our program, this small hotel has 10 rooms. Each room costs \$80 per night. To rent one room, one person must register for the room by giving the following data:

name and the check-in date

We use an “array of objects” to record the data for the customers registered in the rooms. The array has 10 elements (0 to 9), representing room 1 to room 10. Each element of the array is a `Customer` object, or null if the room is available (vacant).

The program consists of three classes: `HotelProgram`, `HotelPanel`, and `Customer`. The main part of the program is shown below, and your task is to complete two methods, as described below.

```
import mcs.comp100.*;
public class HotelProgram {
    public static void main (String[] args) {
        UI window = new UI("Hotel Program");
        new HotelPanel(window);
    }
}

class HotelPanel implements ButtonListener {
    private UI win;
    private Customer[] roomRecord;

    public HotelPanel(UI w) {
        win = w;

        roomRecord = new Customer[10];
        win.addButton("Check In", this);
        win.addButton("Check Out", this);
    }

    public void buttonPerformed(String but) {
        if (but.equals("Check In"))
            checkIn();
        else if (but.equals("Check Out"))
            checkOut();
    }

    private void checkIn() {
    }

    private void checkOut() {
    }
}
```

**(Question 4 continued)**

```
class Customer {
    private String name;
    private String dateCheckIn;

    public Customer(UI u) {
        name = u.getString("Customer name:");
        dateCheckIn = u.getString("Date to check in:");
    }

    public void print(UI u) {
        u.println(" Name: " + name);
        u.println(" Date to check in : " + dateCheckIn);
    }

    public double getCost(int days) {
        return 80.0 * days;
    }
}
```

**(a)** [10 marks] Complete the `checkIn` method for the `HotelPanel` class. This method should have the following steps:

- Prompt for a room number.
- If the room is not available, print a message and return immediately.
- If the room is available, allow the customer to register for the room and store data in the array.
- Print the data for the customer to the screen and say which room he/she will stay in.

```
private void checkIn() {
```

```
}
```

**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.

(b) [10 marks] Complete the following `checkOut` method.

- Prompt for a room number.
- If nobody registered in this room, print a message and return immediately.
- Prompt for how many days the customer has stayed.
- Calculate the cost.
- Print the bill, including data of the customer and the cost.
- Delete the data of this customer in the array, in other words, make the room available.

```
private void checkOut() {
```

```
}
```

## Question 5. Understanding composition

[30 marks]

Consider the following Java classes:

```
import mcs.comp100.*;

class Bird {
    private UI win;
    private int bi;
    public Bird(UI w, int i) {
        win = w;
        bi = i;
    }

    public void chirp() {
        bi++;
        win.print(bi + " ");
    }
}

class Cat {
    private UI win;
    private Bird cb;

    public Cat(UI w, int i) {
        win = w;
        cb = new Bird(win, 10 * i);
    }

    public void purr() {
        cb.chirp();
    }
}

class Dog {
    private UI win;
    private Bird db;

    public Dog(UI w, Bird b) {
        win = w;
        db = b;
    }

    public void bark() {
        db.chirp();
    }

    public Bird fetch(int i) {
        db = new Bird(win, i);
        return db;
    }
}
```

(a) [6 marks] Write what is printed by the program BCD1 below.

```
class BCD1 {
    public static void main(String[] args) {
        UI u = new UI("BCD");
        Bird b1 = new Bird(u, 1);
        Bird b2 = new Bird(u, 20);
        b1.chirp();
        b2.chirp();
        b2.chirp();
    }
}
```

(b) [6 marks] Write what is printed by the program BCD2 below.

```
class BCD2 {
    public static void main(String[] args) {
        UI u = new UI("BCD");
        Cat c1 = new Cat(u, 1);
        Cat c2 = new Cat(u, 20);

        c1.purr();
        c2.purr();
        c2.purr();
    }
}
```

(c) [6 marks] Write what is printed by the program BCD3 below.

```
class BCD3 {
    public static void main(String[] args) {
        UI u = new UI("BCD");
        Bird b3 = new Bird(u, 300);
        Dog d1 = new Dog(u, b3);

        d1.bark();
        d1.fetch(1000).chirp();
        b3.chirp();
    }
}
```

**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.

(d) [6 marks] Write what is printed by the program BCD4 below.

```
class BCD4 {
    public static void main(String[] args) {
        UI u = new UI("BCD");
        Cat[] ca = new Cat[3];

        for (int i = 0; i < 3; i++) {
            ca[i] = new Cat(u, 10 * i);
        }
        for (int i = 2; i >= 0; i--) {
            ca[i].purrr();
        }
    }
}
```

(e) [6 marks] Write what is printed by the program BCD5 below.

```
class BCD5 {
    public static void main(String[] args) {
        UI u = new UI("BCD");
        Dog[] da = new Dog[3];
        Bird b = new Bird(u, 0);

        for (int i = 0; i < 3; i++) {
            da[i] = new Dog(u, b);
            da[i].bark();
            b = da[i].fetch(100 * (i+1));
        }
    }
}
```

## Question 6. Program design with containers

[30 marks]

Consider the following Java program, similar to the mastery exercise in lab assignment 10:

```
import mcs.compl00.*;
public class PersonnelSystem {
    public static void main(String[] args) {
        PersonnelPanel pp = new PersonnelPanel(new UI("Personnel System"));
    }
}

class PersonnelPanel implements ButtonListener {
    private UI win;
    private Staff people;

    public PersonnelPanel(UI w) {
        win = w;
        people = new Staff();
        w.addButton("Add Employee", this);
        w.addButton("Print Employee Details", this);
        w.addButton("Total Salaries", this);
    }

    public void buttonPerformed(String button) {
        if (button.equals("Add Employee"))
            add();
        else if (button.equals("Print Employee Details"))
            print();
        else if (button.equals("Total Salaries"))
            totalSalaries();
    }

    private void add() {
        Employee e = new Employee(win);
        if (people.insert(e))
            win.println("Employee added successfully!");
        else
            win.println("Unable to add employee!");
    }

    private void print() {
        String eid = win.getString("Employee ID?");
        Employee p = people.retrieve(eid);
        if (p != null)
            p.print(win);
        else
            win.println("Unable to retrieve employee with that ID!");
    }

    private void totalSalaries() {
        double total;
        total = people.sumSalaries();
        win.println("Total of all salaries:" + total);
    }
}
```

```

class Employee {
    private String id;
    private String name;
    private double salary;

    public Employee(UI w) {
        id = w.getString("Enter ID:");
        name = w.getString("Enter Name:");
        salary = w.getDouble("Enter Salary:");
    }

    public void print(UI w) {
        w.println("ID: " + id);
        w.println("  Name: " + name);
        w.println("  Salary: " + salary);
    }

    public String getID() {
        return id;
    }

    public double getSalary() {
        return salary;
    }
}

//Staff is not shown here in detail:
// insert: is to store an employee record,
//         and returns true if it was able to, false otherwise
//         (e.g. duplicate record or no more storage space).
// retrieve: is to return an employee record matching an ID,
//           and returns null if there is not one that matches.
// sumSalaries returns the total of all employee salaries.

class Staff {
    public Staff() {...}
    public boolean insert(Employee x) {...}
    public Employee retrieve(String id) {...}
    public double sumSalaries() {...}
}

```

**(a)** [3 marks] List the names of the methods of the `PersonnelPanel` class that are called when the Add Employee button is pressed.

**(b)** [3 marks] List the names of the methods of the `Staff` class that are called by methods of the `PersonnelPanel` class.

**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.

(c) [12 marks]

Assume that the Staff class has fields, constructor, and an insert method as shown below. Show what the code would be for the retrieve and sumSalaries methods.

```
class Staff {
    private Employee[] workers;
    private int maxsize = 100;
    private int size = 0;

    public Staff() {
        workers = new Employee[maxsize];
    }

    public boolean insert(Employee x) {
        if (size == maxsize)
            return false;
        workers[size] = x;
        size++;
        return true;
    }
}
```

```
public Employee retrieve(String id) {

}
}
```

```
public double sumSalaries() {

}
}
```

```
}
```

**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.

(d) [12 marks]

Assume you have a generic Map class with the following methods:

```
public boolean insert(Object k, Object v)
    // returns true if insert ok, false if duplicate or storage full
public Object lookup(Object key)
    // returns value matching key, or null if none found
public void getInit()    // initialise for traversal
public boolean getNext() // go to next position in traversal
public Object getKey()  // get key at current position in traversal
public Object getValue() // get value at current position in traversal
```

An alternative implementation for the Staff class would be to use the generic Map class from inside the Staff class. Assume that the Staff class has fields, constructor, and an insert method as shown below. Show what the code would be for the retrieve and sumSalaries methods.

```
class Staff {
    private Map workers;

    public Staff() {
        workers = new Map();
    }

    public boolean insert(Employee x) {
        boolean res;
        res = workers.insert(x.getID(), x);
        return res;
    }
```

```
public Employee retrieve(String id) {

}
```

```
public double sumSalaries() {

}
```

```
}
```

## Question 7. Recursion

[12 marks]

Consider the following `calculate` method:

```
1 public int calculate(int n){
2     Trace.println(n);
3     if ( n <= 0 )
4         return 0;
5     else if ( n <= 3 )
6         return n;
7     else {
8         int num1 = calculate(n-1);
9         int num2 = calculate(n-2);
10        int num3 = calculate(n-3);
11        return num1 + num2 + num3;
12    }
13 }
```

(a) [2 marks] Write the line numbers of the statements that describe the base cases (stopping cases).

(b) [2 marks] Write the line numbers of the statements that describe the recursive steps.

(c) [4 marks] Complete the following table. The first row gives the value of the argument `n`, the second row gives the value of `calculate(n)`, which is the returned value of calling the `calculate` method with argument `n`.

n	0	1	2	3	4	5	6	7
calculate(n)	.....	.....	.....	.....	.....	.....	.....	.....

(d) [4 marks] What will be printed out in the Trace window if this method is called with an argument of 5?

\*\*\*\*\*

**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.

## Documentation.

### Methods for input and output with UI.

<code>public void <u>print</u>(..)</code>	<i>Print a value in the text pane.</i>
<code>public void <u>println</u>(. . .)</code>	<i>Print a value in the text pane, and start a new line.</i>
<code>public int <u>getInt</u>(String prompt)</code>	<i>Ask the user for an integer.</i>
<code>public double <u>getDouble</u>(String prompt)</code>	<i>Ask the user for a double.</i>
<code>public String <u>getString</u>(String prompt)</code>	<i>Ask the user for a string of characters.</i>
<code>public boolean <u>getBoolean</u>(String prompt)</code>	<i>Ask the user for a yes/no answer, and return true or false.</i>
<code>public boolean <u>getChar</u>(String prompt)</code>	<i>Ask the user for a single character.</i>

### Event Driven Input with the UI. Assume win is of type UI.

<code>public void <u>addButton</u>(String but, ButtonListener obj)</code>	<i>Add button named but to window. Call <code>buttonPerformed</code> on obj when the button is clicked</i>
<code>public void <u>addTextField</u>(String tf, TextFieldListener obj)</code>	<i>Add text-field named tf to window. Call <code>textFieldPerformed</code> on obj when value entered in the field</i>
<code>public void <u>addNumberField</u>(String nf, NumberFieldListener obj)</code>	<i>Add number-field named nf to window. Call <code>numberFieldPerformed</code> on obj when value entered</i>
<code>public void <u>addSlider</u>(String sl, double min, double max, SliderListener obj)</code>	<i>Add a number-field named sl to the window with range of values from min to max. Call <code>sliderPerformed</code> on obj when slider is moved</i>
<code>public void <u>buttonPerformed</u>(String but)</code>	<i>A class that implements <code>ButtonListener</code> must provide this method. It is called if a button is clicked. but is the name of the button that was clicked.</i>
<code>public void <u>textFieldPerformed</u>(String tf, String val)</code>	<i>A class that implements <code>TextFieldListener</code> must provide this method. It is called if a value is entered in a text-field. tf is the name of the field and val is the value that was entered.</i>
<code>public void <u>numberFieldPerformed</u>(String nf, double val)</code>	<i>A class that implements <code>NumberFieldListener</code> must provide this method. It is called if a value is entered in a number-field. nf is the name of the field and val is the value that was entered.</i>

### Operators:

<code>x + y</code>	<i>if x and y are both numbers, returns the sum of x and y if either x or y is a string, changes the other to a string and concatenates them into a single string</i>
<code>x - y</code>	<i>x and y must both be numbers; returns the difference of x and y</i>
<code>x++</code>	<i>Makes the value in x larger by 1, and returns the original value of x</i>
<code>x--</code>	<i>Makes the value in x smaller by 1, and returns the original value of x</i>
<code>x * y</code>	<i>x and y must both be numbers; returns the product of x and y</i>
<code>x / y</code>	<i>if x and y are both integers, returns the integer quotient of x and y if either x or y is a double, returns the real quotient of x and y</i>
<code>x % y</code>	<i>x and y must both be numbers, returns the remainder of <math>x \div y</math></i>

## Reading from files:

<code>new <u>DataFileReader</u>();</code>	<i>Asks the user to select a file, and constructs a new <code>DataFileReader</code> object that will read data from that file.</i>
<code>public boolean <u>endOfFile</u>()</code>	<i>Returns true if the <code>DataFileReader</code> is at the end of the file.</i>
<code>public void <u>close</u>()</code>	<i>Closes the file. It is an error to read from it again.</i>
<code>public void <u>reset</u>()</code>	<i>Resets to the beginning of the file.</i>
<code>public String <u>readLine</u>()</code>	<i>Reads and returns a string containing the next line of characters from the file. Returns <b>null</b> if no more lines in file.</i>
<code>public String <u>readToken</u>()</code>	<i>Reads and returns a string containing the next group of non-blank characters from the file. Could be a word, a string of digits, or a sequence of any characters (except space and the end-of-line character). Returns <b>null</b> if no more non-blank characters in the file.</i>
<code>public double <u>readNumber</u>()</code>	<i>If the next token (sequence of non-blank characters) is a number, reads and returns the number represented by those digits. If the next token is not a number, then it does not read anything, and returns the special value <code>Double.MAX_VALUE</code>.</i>

## Methods on Strings:

<code>public int <u>length</u>()</code>	<i>Returns the length of this string</i>
<code>public boolean <u>equals</u>(String str)</code>	<i>Returns true if and only if <code>str</code> and this string contain the same characters.</i>
<code>public String <u>substring</u>(int i1, int i2)</code>	<i>Returns a new string that is a substring of this string, beginning at <code>i1</code> and extending to the index <code>(i2-1)</code>. An index ranges from 0 to <code>(length()-1)</code>. e.g., <code>"Otaki".substring(1, 4)</code> returns a string containing just the characters <code>"tak"</code>.</i>
<code>public char <u>charAt</u>(int index)</code>	<i>Returns the character at the specified index. An index ranges from 0 to <code>length()-1</code>.</i>
<code>public String <u>concat</u>(String str)</code>	<i>Returns a new string consisting of the characters in this string followed by the characters in the argument string.</i>

## The Math class:

<code>Math.PI</code>	<i>a constant with the value <math>\pi</math> 3.14159...</i>
<code>public double <u>sqr</u>(double n)</code>	<i>Returns the square of <code>n</code>.</i>
<code>public double <u>sqrt</u>(double n)</code>	<i>Returns the square root of <code>n</code>.</i>
<code>public double <u>abs</u>(double n)</code>	<i>Returns the absolute value of <code>n</code>.</i>
<code>public double <u>min</u>(double m, double n)</code>	<i>Returns the smaller of <code>m</code> and <code>n</code>.</i>
<code>public f double <u>max</u>(double m, double n)</code>	<i>Returns the larger of <code>m</code> and <code>n</code>.</i>

## The Trace class:

<code>public void <u>setVisible</u>(boolean f)</code>	<i>Set the trace window visible or hidden. if the argument is true, the trace window will be visible.</i>
<code>public void <u>println</u>(...)</code>	<i>Print a value to the Trace window and start a new line.</i>