Changing your password

The initial password is random, so it might not be easy to remember the initial password.

1. Click at the Start button, and choose "Run".
2. When the dialog box appears, type "telnet virtue". This allows you to directly access the server of our department.
3. A new window appears. Type in your user name and your (old) password.
4. A line appears in the window: "TERM = (vt100) ". Type enter. We will not use the terminal in full-screen, so anything works.
5. Type "passwd". It will asks you for your old password and then your new password. You also have to retype your new password to confirm the change. Now the password is changed.
6. Close the window. Logout yourselves, and try to use the new password to login.

Forwarding mail

You can receive mails in your computer accounts of the CSIS department. The mail address is "username@csis.hku.hk". However, it is troublesome to check mails in many accounts.

A solution is to forward all the mails to the mail account that you usually use. Suppose you usually use h0012345@hkusub.hku.hk. Then you can do the following to forward the mails:

1. Run the notepad program.
2. Type the mail address, `h0012345@hkusua.hku.hk`, that you usually use.
3. Save it in `H:\.forward`. The forwarding setup is done.
4. Try to send a mail to your CSIS account, and see whether you can receive it in your mailbox (i.e. h0012345@hkusua.hku.hk in this case).

Java installation in a nutshell

To install Java at your own computers is a little bit easier than to do it in our lab, since you can change anything of your own computer.

1. Get the CD. (You might need to borrow it from your classmates.)
2. Place the CD into the computer. Double click on the "My Computer" icon and then the CD icon.
3. Double click the j2sdk... icon. This starts the installation of Java. Accept all defaults.
4. Double click the chapman_... icon. This starts WinZip. Extract the files to a directory, say `C:\bookpkg`.
5. Run notepad, and open the file `C:\Autoexec.bat`.
6. Add a line `set CLASSPATH=C:\bookpkg;.`. Modify the line starting like "PATH=" or "set PATH=", by adding `;C:\JDK1.3\bin` at the end. Reboot.

Revision: if/else statement

- By using the if/else statement, we can achieve data dependent decisions in Java.
- The general form if an if/else statement contains an if part and an else part, each part containing a single statement.
- When the else part is not needed, you can omit it completely.
- When the if part is not needed, you can write a null statement there.
- When you need to place multiple statements in a part, group them together using braces {} to form a single compound statement.

Nested if/else statements

- The if/else statement is one single statement.
- Being a statement, it can be part of another if/else statement.
- If an if/else statement lies inside another if/else statment, it is said to be "nested".
- Example:

```
if (Condition1)
  if (Condition2)
    statement1
  else
    statement2
else
  statement3
```


Nested if/else statements (cont.)

- Arbitrary nesting is allowed. However, if the nesting goes more than a few levels, it becomes very difficult to understand the program.
- Nested if/else is usually used to achieve multiple branch. It is of the following form:

```
if (Condition1)
  statement1
else
  if (Condition2)
    statement2
  else
    if (Condition3)
      statement3
    else
      statement4
```

or

```
if (Condition1)
  statement1
else if (Condition2)
  statement2
else if (Condition3)
  statement3
else
  statement4
```

Example: leap year

Question: given a year number (e.g. 2000), is it a leap year?

Answer: a year is a leap year if it is divisible by 400. Otherwise, it is a not leap year if it is divisible by 100. Otherwise, it is a leap year if it is divisible by 4. Otherwise it is not a leap year.

Program (more interesting part):

```
System.out.print("Year? ");
int year = in.readInt();
boolean is_leap;
if (year % 400 == 0)
  is_leap = true;
else if (year % 100 == 0)
  is_leap = false;
else if (year % 4 == 0)
  is_leap = true;
else
  is_leap = false;
if (is_leap)
  System.out.println(year + " is a leap year.");
else
  System.out.println(year + " is not a leap year.");
```


Nested if/else statements: Simpler things first

We normally want to write the simpler part as the if part, leaving the more complicated part as the else part.

Example: what will the last program looks like if this rule is violated?

```
System.out.print("Year? ");
int year = in.readInt();
boolean is_leap;
if (year % 400 != 0)
  if (year % 100 != 0)
    if (year % 4 != 0)
      is_leap = false;
    else
      is_leap = true;
  else
    is_leap = false;
else
  is_leap = true;
if (is_leap)
  System.out.println(year + " is a leap year.");
else
  System.out.println(year + " is not a leap year.");
```


The Dangling else problem

The allowance of a missing else part can cause some problem. Consider the following program:

```
import chapman.io.*;
public class HotCold {
  public static void main(String[] args) {
    StdIn in = new StdIn();
    System.out.print("What is the temperature in Celsius? ");
    double temp = in.readDouble();
    if (temp > 30)
      if (temp > 80)
        System.out.println("Too hot!");
    else
      System.out.println("Too cold!");
  }
}
```


The Dangling else problem (cont'd)

- Note that the program makes sense either if the else statement is attached to the first if or the second if statement.
- This happens when the inner if statement is not matched with an else statement but the outer if statement is.
- The rule: an else statement is always attached to the nearest unterminated if statement.
- That means, the else statement is attached to the **second** if, not the first if.
- Note that even if the indentation suggested that the programmer want the else to attach to the first if, the compiler will not see it this way. I.e., the program will not do what the programmer expects.

Solution 1: Use more else

One of the solution is to terminate all the if/else statements. This sometimes results in a lot of null-statement, and so the resulting program may not be very nice looking.

```
import chapman.io.*;
public class HotCold {
  public static void main(String[] args) {
    StdIn in = new StdIn();
    System.out.print("What is the temperature in Celsius? ");
    double temp = in.readDouble();
    if (temp > 30)
      if (temp > 80)
        System.out.println("Too hot!");
      else
        ;
    else
      System.out.println("Too cold!");
  }
}
```


Solution 2: Use compound statement

Another solution is to force the compiler into the correct interpretation by making the inner if statement the content of a compound statement. Example:

```
import chapman.io.*;
public class HotCold {
  public static void main(String[] args) {
    StdIn in = new StdIn();
    System.out.print("What is the temperature in Celsius? ");
    double temp = in.readDouble();
    if (temp > 30) {
      if (temp > 80)
        System.out.println("Too hot!");
    }
    else
      System.out.println("Too cold!");
  }
}
```

Program design

When a program becomes more and more complicated, it becomes very easy to get lost in small details, forgetting about the big problem.

A formal procedure for writing programs is important when writing bigger programs.

We design our programs with the **Top down design** technique: we try to view our problems in a very high level, skipping all details at first.

When we have the skeleton of the program, and when we believe to a certain degree that the high level view is correct, we start thinking about the details one by one.

We test our programs at every place of our design process, not just when the whole program is ready.

The program design process

1. State the problem.
2. Define the input and output.
3. Design a step by step procedure to produce the required output, skipping details that you don't want to tackle immediately.
4. Convert the procedure into Java code.
5. Test your program.
6. If there is any details that is not completed, go back to (3).
7. Test your program again.

Example: solving a triangle

Suppose we want a program that calculates all angle and sides of a triangle, given any two angles and any side.

This is clearly possible: once we have two angles we have the third.

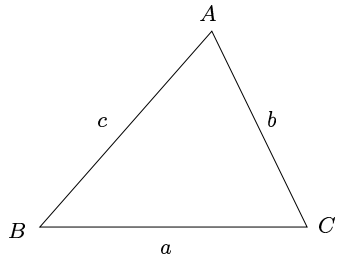
Our primary weapons:

Total angle of triangle:

$A + B + C = 180^\circ$

and the sine rule:

$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$



The problem, the input and output

Problem: Given two of A, B and C, and given one of a, b and c, find all of them.

Input: Two of A, B and C. The user is asked about them one by one, and is permitted to type 0 if it is unknown. The input of C should be skipped if both A and B are known.

One of a, b and c. The user is asked about them one by one, and is permitted to type 0 if it is unknown. The program should stop asking once a side is entered.

Output: One of:

- the values of A, B, C, a, b and c; or
- a message indicating that it is not a triangle.
- a message indicating that there is not enough information.

The overall procedure

Ask for the three angles, don't ask for C if we have both A and B.
If less than 2 angles are given, print an error message.
Else
    Find all the angles.
    If any angle is not positive, print an error message.
    Else
        Ask for the three sides, until we get a non-zero value.
        If no side is given, print an error message.
        Else
            Find all sides.
            Print the sides and angles.

Turning it to Java

```
/* Solve a triangle given two angles and one side */
import chapman.io.*;
public class SolveTriangle {
  public static void main(String[] args) {
    double a, b, c, A, B, C;
    StdIn in = new StdIn();

    System.out.println("FIXME: Should ask for two angles.");
    if (false) // FIXME: if less than two angles are given
      System.out.println("Not enough angles!");
    else {
      System.out.println("FIXME: Should calculate all angles.");
      if (false) // FIXME: if any angle is non-positive
        System.out.println("Some angle is not positive.");
      else {
        System.out.println("FIXME: Should ask for a side");
        if (false) // FIXME: if no side is given
          System.out.println("No side is given!");
        else {
          System.out.println("FIXME: Should find all sides.");
          System.out.println("FIXME: Should print all sides and angles.");
        }
      }
    }
  }
}
```

Why writing non-working code?

- The code above will only print out a lot of "FIXME" messages.
- But this step is important: it reminds you what is your initial design.
- And the FIXME place-holders allow you to check whether the program compile right and run correctly.
- Once you are convinced that the program work correctly (e.g. by changing the boolean values and re-run), you start to actually refine the program by "fixing" the missing parts.

Getting the angles

Procedure:
Ask for A.
Ask for B.
If at least one of A and B is 0
    Ask for C.
Else
    Set C to 0.

Program:

```
// Ask for the angles
System.out.print("A? (0 if unknown) ");
A = in.readDouble();
System.out.print("B? (0 if unknown) ");
B = in.readDouble();
if (A==0 || B==0) {
  System.out.print("C? ");
  C = in.readDouble();
}
else
  C = 0;
```


Checking the angles; finding all angles

If two of the angles are 0, then we have insufficient information. In java, we just need to check (A==0 && B==0), (B==0 && C==0) and (A==0 && C==0). If any one is true, then we have an error.

To find the remaining angle: subtract 180 from the remaining angles. I.e. if A is missing, A is $180 - B - C$. Similar for B and C.

```
// Calculate all angles
if ((A==0 && B==0) || (B==0 && C==0) || (C==0 && A==0)) //
  System.out.println("Not enough angles!");
else {
  if (A==0)
    A = 180 - B - C;
  else if (B==0)
    B = 180 - A - C;
  else
    C = 180 - A - B;
  if (A<=0 || B<=0 || C<=0)
    System.out.println("Some angle is not positive");
  else {
    ...
```


Early testing

Testing your program early. This way problems are revealed earlier when there is less things that can go wrong.

The following prints the values of the angle so that you can be sure things are going right.

```
System.out.println("A = " + A);
System.out.println("B = " + B);
System.out.println("C = " + C);
```

Example output

```
A? (0 if unknown) 0
B? (0 if unknown) 90
C? 60
A = 30.0
B = 90.0
C = 60.0
FIXME: Should ask for a side
FIXME: Should find all sides.
FIXME: Should print all sides and angles.
```


Asking for a side, and checking it

Procedure
Ask for a.
If a is non-zero
    Set b and c to 0.
Else
    Ask for b.
    If b is non-zero
        Set c to 0.
    Else
        Ask for c.
If all a, b and c are 0
    Error: no side given.

Program

```
// Ask for a side
System.out.print("a? (0 if unknown) ");
a = in.readDouble();
if (a != 0)
  b = c = 0;
else {
  System.out.print("b? (0 if unknown) ");
  b = in.readDouble();
  if (b != 0)
    c = 0;
  else {
    System.out.print("c? ");
    c = in.readDouble();
  }
}
if (a==0 && b==0 && c==0)
  System.out.println("No side is given!");
...
```


Finding the sides

Now it comes to the real computation. Note that it is only a small part of the problem, and it require much less code than the remainder of the program.

Procedure:
If a is given
    Ratio = a/sin(A)
Else If b is given
    Ratio = b/sin(B)
Else
    Ratio = c/sin(C)
a = Ratio * sin(A)
b = Ratio * sin(B)
c = Ratio * sin(C)
Print everything

Program:

```
// Find the sides
double ratio;
if (a != 0)
  ratio = a / Math.sin(A*Math.PI/180);
else if (b != 0)
  ratio = b / Math.sin(B*Math.PI/180);
else
  ratio = c / Math.sin(C*Math.PI/180);
a = ratio * Math.sin(A*Math.PI/180);
b = ratio * Math.sin(B*Math.PI/180);
c = ratio * Math.sin(C*Math.PI/180);
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("A = " + A);
System.out.println("B = " + B);
System.out.println("C = " + C);
```