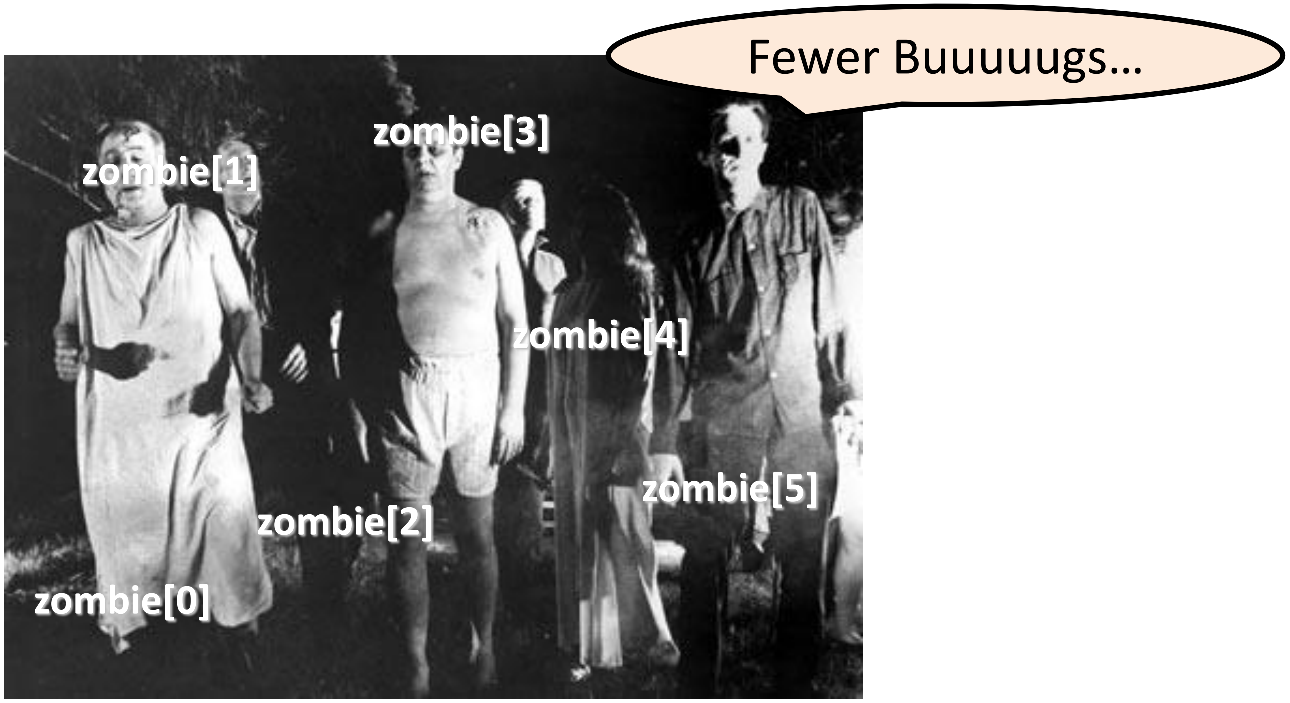
Problem Decomposition: One Professor's Approach to Coding



Overview

- Problem Solving
  - Understand the Problem
  - Work out the Logic
  - Convert it to Code
  - We will walk through a sample problem and go through these steps

The (Example) Problem

- You are a communications officer onboard an E-3 Sentry AWACS surveillance aircraft. Your radar equipment generates a file containing all radar contacts within your region. Each radar contact is given by its UTM coordinates. UTM coordinates consist of two numbers, an easting and a northing. The easting specifies how many meters the contact is to the east of a fixed grid reference location. Similarly the northing is how far the contact is north of the grid reference.

The (Example) Problem

In addition to the location of every aircraft, your equipment also queries the transponder of all aircraft to obtain their call sign. Only friendly aircraft respond with a call sign. Other unknown or hostile aircraft are assigned a question mark as a call sign. Here is radar4.txt, a small example file showing four contacts, two friendly and two unknown:

```
4
34754 12029 EJ-475
38002 11193 CX-120
11899 28929 ?
39222 10028 ?
```

The (Example) Problem

The first line of the example file specifies that there are four contacts in the data file. Each of the remaining lines gives the easting, northing and call sign (in that order). You can assume all eastings and northings are non-negative integers.

The (Example) Problem

- Your job is to write a program RadarContacts.java that first reports the number of friendly and non-friendly contacts in the region. The program then generates a list of warning messages for transmission to all friendly aircraft. The messages inform each friendly aircraft about the distance to any radar contacts that are too close. You program should take three command-line arguments radius, mode and the name of the file that contains the data. The radius argument specifies how close (in kilometers) another contact must be before a warning is generated. A warning should be generated if the two-dimensional Euclidean distance between the friendly aircraft in question and the contact is less than or equal to radius kilometers. The radius argument can be any non-negative floating-point value (e.g. 1.5).

The (Example) Problem

- The mode command-line argument specifies the type of contacts that should be reported:
  - mode 0, only non-friendly contacts are listed
  - mode 1, only friendly contacts are listed
  - mode 2, first non-friendly contacts are listed, followed by friendly contacts

The (Example) Problem

- The output report format first lists the call sign of the friendly aircraft followed by a colon. In mode 0 and mode 2, all non-friendly that are too close are listed (denoted by a question mark) followed by the bogey's distance (in kilometers) in parentheses. In mode 1, all friendly contacts are listed by their known call sign followed by the friendly's distance in parentheses. If a friendly contact does not have any contacts that are too close (base on the mode), it should not appear in the report. All distances should be reported in kilometers rounded to two decimal places. Contacts within the friendly and non-friendly sets for a given aircraft's report can appear in any order.

The (Example) Problem

- Here are some example runs:

```
% java RadarContacts 1.0 2 radar4.txt
Friendly aircraft: 2
Non-friendly aircraft: 2

% java RadarContacts 2.0 2 radar4.txt
Friendly aircraft: 2
Non-friendly aircraft: 2
CX-120:
? (1.69)
```

The (Example) Problem

- Here are some (more) example runs:

```
% java RadarContacts 3.5 2 radar4.txt
Friendly aircraft: 2
Non-friendly aircraft: 2
EJ-475:
CX-120 (3.35)

CX-120:
? (1.69)
EJ-475 (3.35)
```

The (Example) Problem

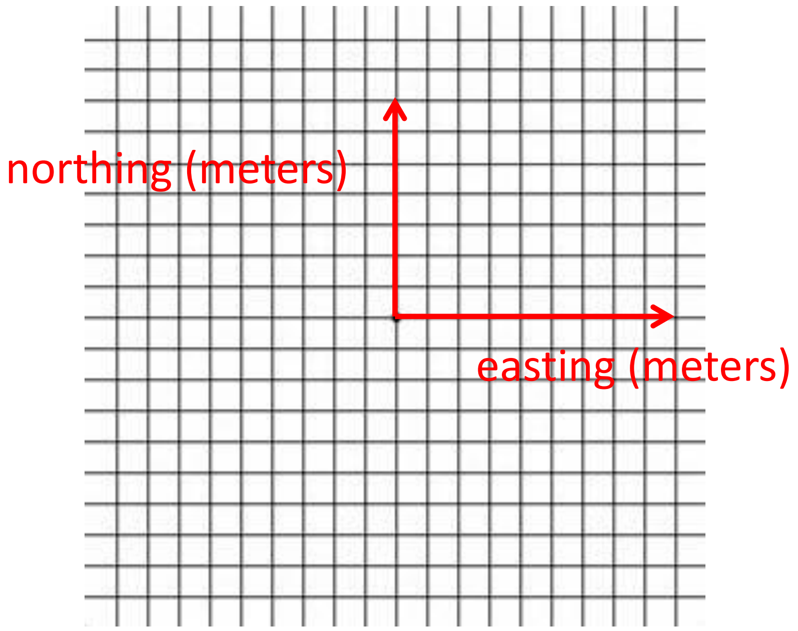
- Here are some (more) example runs:

```
% java RadarContacts 3.5 1 radar4.txt
  Friendly aircraft: 2
  Non-friendly aircraft: 2
  EJ-475:
  CX-120 (3.35)

  CX-120:
  EJ-475 (3.35)

  % java RadarContacts 3.5 0 radar4.txt
  Friendly aircraft: 2
  Non-friendly aircraft: 2
  CX-120:
  ? (1.69)
```

Understanding the Problem

- You are a communications officer onboard an E-3 Sentry AWACS surveillance aircraft. Your radar equipment generates a file containing all radar contacts within your region. Each radar contact is given by its UTM coordinates. UTM coordinates consist of two numbers, an easting and a northing. The easting specifies how many meters the contact is to the east of a fixed grid reference location. Similarly the northing is how far the contact is north of the grid reference.

OK – I know I have two numbers in here, an easting and a northing. They are in meters, and they tell me how far east and north of a fixed point on a grid something is…



Understanding the Problem

In addition to the location of every aircraft, your equipment also queries the transponder of all aircraft to obtain their call sign. Only friendly aircraft respond with a call sign. Other unknown or hostile aircraft are assigned a question mark as a call sign. Here is radar4.txt, a small example file showing four contacts, two friendly and two unknown:

```
  4
34754 12029 EJ-475
38002 11193 CX-120
11899 28929 ?
39222 10028 ?
```

The data I'm going to get is in a file. In addition to easting and northing I have a call sign that looks like it's a String data type. Friendlies will give me a call sign, and unfriendlies are shown as "?".

Understanding the Problem

The first line of the example file specifies that there are four contacts in the data file. Each of the remaining lines gives the easting, northing and call sign (in that order). You can assume all eastings and northings are non-negative integers.

OK, that first number in the file tells me how many contacts will be listed in the file. And the easting and northing numbers will all be non-negative integers.

Understanding the Problem

- Your job is to write a program RadarContacts.java that first reports the number of friendly and non-friendly contacts in the region. The program then generates a list of warning messages for transmission to all friendly aircraft. The messages inform each friendly aircraft about the distance to any radar contacts that are too close. Your program should take three command-line arguments, radius, mode and a file name. The radius argument specifies how close (in kilometers) another contact must be before a warning is generated. A warning should be generated if the two-dimensional Euclidean distance between the friendly aircraft in question and the contact is less than or equal to radius kilometers. The radius argument can be any non-negative floating-point value (e.g. 1.5).

That's a lot of stuff! Let's sort it all out…

I have to report the number of friendlies and nonfriendlies. That means I'll have to count them both.

I have to generate a list of warning messages to friendlies about contacts that are too close, using Euclidean distance.

I have to take in three command line arguments – one that tells me how close is too close, the radius. It will be in kilometers. And it will be a non-negative floating point number.

The other command line argument is something called a "mode".

And finally, the file name.

Understanding the Problem

- The mode command-line argument specifies the type of contacts that should be reported:
  - mode 0, only non-friendly contacts are listed
  - mode 1, only friendly contacts are listed
  - mode 2, first non-friendly contacts are listed, followed by friendly contacts

Ah, there's where "mode" is explained. It's either a 0, a 1, or a 2. And it tells my program which "too-close" contacts are to be reported.

Understanding the Problem

- The output report format first lists the call sign of the friendly aircraft followed by a colon. In mode 0 and mode 2, all non-friendly that are too close are listed (denoted by a question mark) followed by the bogey's distance (in kilometers) in parentheses. In mode 1, all friendly contacts are listed by their known call sign followed by the friendly's distance in parentheses. If a friendly contact does not have any contacts that are too close (base on the mode), it should not appear in the report. All distances should be reported in kilometers rounded to two decimal places. Contacts within the friendly and non-friendly sets for a given aircraft's report can appear in any order.

But wait! There's more!
I have to output the call sign of the friendly aircraft followed by a colon.
Then if I'm in mode 0 or 2 I have to list the nonfriendlies by a "?"
followed by their distance in parentheses.
If I'm in mode 1 I print the friendlies. I think that should really be mode 1
or 2.
Only friendlies that have "too close" contacts should be output.
Distances should be in kilometers.
Distances should be rounded to two decimal places.

Understanding the Problem

- Here are some example runs:

```
% java RadarContacts 1.0 2  radar4.txt
Friendly aircraft: 2
Non-friendly aircraft: 2

% java RadarContacts 2.0 2  radar4.txt
Friendly aircraft: 2
Non-friendly aircraft: 2
CX-120:
? (1.69)
```

Ah, so this is what the output should look like. The count of the friendlies and non-friendlies. Then a call sign for a friendly (if there are any with “too close” contacts), followed by a colon, then the call sign of the “too close” contact and a distance in parentheses.

Understanding the Problem

- Here are some (more) example runs:

```
% java RadarContacts 3.5 2  radar4.txt
Friendly aircraft: 2
Non-friendly aircraft: 2
EJ-475:
CX-120 (3.35)

CX-120:
? (1.69)
EJ-475 (3.35)
```

Another example with a larger radius. This time it looks like there are two friendlies that have “too close” contacts.

Understanding the Problem

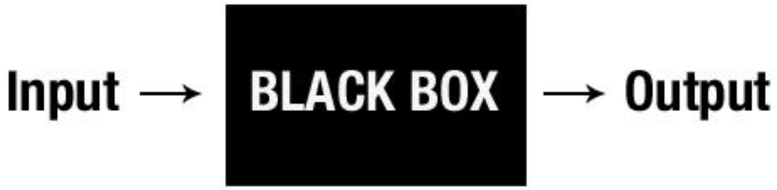
- Here are some (more) example runs:

```
% java RadarContacts 3.5 1  radar4.txt
   Friendly aircraft: 2
   Non-friendly aircraft: 2
   EJ-475:
   CX-120 (3.35)

   CX-120:
   EJ-475 (3.35)

   % java RadarContacts 3.5 0  radar4.txt
   Friendly aircraft: 2
   Non-friendly aircraft: 2
   CX-120:
   ? (1.69)
```

OK, and a couple of examples where the mode is not 2.

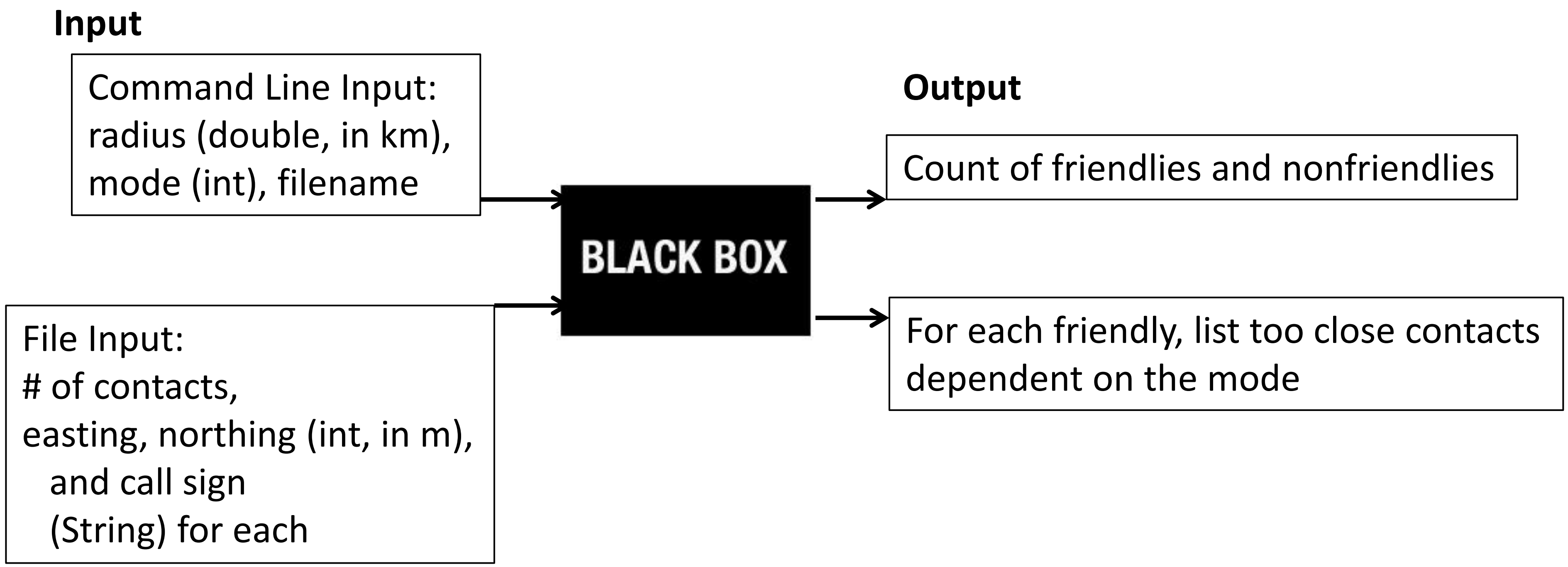
Working out the Logic

- OK. I think I have a better understanding of the pieces of the problem, but I still don’t know how to write a program to do all that.
- My next step will be to sort out the problem pieces so that I can figure out the logic
- Toward the beginning of the semester, we talked about a program being a black box, where input goes into it, and is transformed into whatever it’s supposed to output.
  - Let’s look at that part first…



Working out the Logic

- I have three command line inputs, the radius, the mode and a file name
- I have a file input that tells me the number of contacts and the easting, northing and call sign of each
- I have two parts to the output
  - First, I have to output the count of the friendlies and the nonfriendlies
  - Then I have to output the “report” – for each friendly, I have to list any contacts that are “too close”, dependent on the mode that was input on the command line


Input
Command Line Input: radius (double, in km), mode (int), filename
File Input: # of contacts, easting, northing (int, in m), and call sign (String) for each
BLACK BOX
Output
Count of friendlies and nonfriendlies
For each friendly, list too close contacts dependent on the mode

Working out the Logic

- I think I can start working on the logic, just given the inputs and outputs.
  - Here's a start…

```
Get radius, mode, filename from command line.
Get number of contacts from file.
For each contact,
        Get easting, northing, call sign.
Count number of nonfriendlies.
Calculate number of friendlies.
Output these to the screen.
```

- Hey, this is great! I think I can code this part, and I've covered ¾ of the program!

- … But wait. There's still that little tricky detail of the output report…
  - OK, let's keep going…

Working out the Logic

- OK, now for a pass at the logic for the report part…

```
Get radius, mode, filename from command line.
Get number of contacts from file.
For each contact,
        Get easting, northing, call sign.
Count number of nonfriendlies.
Calculate number of friendlies.
Output these to the screen.
For each friendly,
        Calculate the distance from it to each other contact.
        If distance < radius
                If mode is 0 or 2, and contact is nonfriendly,
                        Write info to a nonfriendly String
                Else, if mode is 1 or 2 and contact is friendly,
                        Write info to a friendly String.
        If friendly or nonfriendly isn't empty,
                Print call sign followed by a colon.
                Print nonfriendly string.
                Print friendly string.
```

Working out the Logic

```
Get radius, mode from command line.
Get number of contacts from file.
For each contact,
        Get easting, northing, call sign.
Count number of nonfriendlies.
Calculate number of friendlies.
Output these to the screen.
For each friendly,
        Calculate the distance from it to each other contact.
        If distance < radius
                If mode is 0 or 2, and contact is nonfriendly,
                        Write info to a nonfriendly String
                Else, if mode is 1 or 2 and contact is friendly,
                        Write info to a friendly String.
        If friendly or nonfriendly isn't empty,
                Print call sign followed by a colon.
                Print nonfriendly string.
                Print friendly string.
```

- I think that I've worked out the logic for the program, I think that will work. Now for the real test… I'll see if I have glossed over anything important, or made some logic errors when I try to write the code.

Converting it to Code

```
Get radius, mode from command line.
Get number of contacts from file.
For each contact,
        Get easting, northing, call sign.
Count number of nonfriendlies.
Calculate number of friendlies.
Output these to the screen.
For each friendly,
        Calculate the distance from it to each other contact.
        If distance < radius
                If mode is 0 or 2, and contact is nonfriendly,
        Write info to a nonfriendly String
                Else, if mode is 1 or 2 and contact is friendly,
        Write info to a friendly String.
        If friendly or nonfriendly isn't empty,
                Print call sign followed by a colon.
                Print nonfriendly string.
                Print friendly string.
```

```java
public class RadarContacts
{
    public static void main(String [] args)
    {
        Scanner file;
        try
        {
            // Read the command line arguments
            double radius = Double.parseDouble(args[0]);
            int mode = Integer.parseInt(args[1]);
            file = new Scanner(new File(args[2]));

            // Read in the number of contacts and set up arrays
            //   for the easting, northing and call signs
            int contacts = file.nextInt();
            double [] east = new double[contacts];
            double [] north = new double[contacts];
            String [] callSign = new String[contacts];

            // Set up variables for counting friendlies and nonfriendlies
            int friendlies = 0;
            int nonfriendlies = 0;
            String closeFriendlies;
            String closeNonfriendlies;

            // Read all the numbers in and count the numbers of each
            for(int i = 0; i < contacts; i++)
            {
                east[i] = file.nextInt() / 1000.0;
                north[i] = file.nextInt() / 1000.0;
                callSign[i] = file.nextLine();
                if (callSign[i].equals(" ?"))
                {
                    nonfriendlies++;
                }
                else
                {
                    friendlies++;
                }
            }
            // Print out number of friendlies and nonfriendlies
            System.out.println("Friendly aircraft: " + friendlies);
            System.out.println("Non-friendly aircraft: " + nonfriendlies);
 …
```

And here's some code that brings us to the point of reading contacts and counting friendlies and non-friendlies.

Summary

- Problem Solving
  - Understanding the Problem
  - Working out the Logic
  - Converting it to Code