Chapter 1 is where we will start.

I will attempt to use an analogy between (1) a cooking recipe and a kitchen, and (2) a program and a computer.

First let’s look at a cooking recipe:
Recipe (page 2)

ENCHILADAS

1. Preheat oven to 350°F.
2. In a heavy saucepan
   1. Heat 2 tablespoons olive oil.
   2. Sauté
      \[\text{loop}\]
      \[
      \frac{1}{2} \text{ cup chopped onion} \\
      1 \text{ minced garlic clove} \\
      \text{Until golden}
      \]
   3. Add
      \[
      1 \text{ tablespoon chili powder} \\
      1 \text{ cup tomato puree} \\
      \frac{1}{2} \text{ cup chicken stock}
      \]
4. Season with
   Salt and pepper
   1 teaspoon cumin
3. Spread sauce over tortillas.
4. Fill centers with equal quantities of
   Chopped raw onion
   Chopped mozzarella cheese
5. Roll tortillas.
6. Place in ovenproof dish.
7. Pour more sauce over tops.
8. Sprinkle with chopped mozzarella cheese.
9. Heat thoroughly in oven about 15 minutes.

Notes:
- Some steps sequential, some parallel.
- Combine steps into blocks.
- Has loops or iteration.
- Has “conditional”.

Figure 1.1
WALLPAPER

1. input corna, cornb
2. input side
3. for i <- 1 to 100
   1. for j <- 1 to 100
      1. x <- corna + i * side/100
      2. y <- cornb + j * side/100
      3. c <- int( x*x + y*y )
      4. if c is even then plot( i,j )

Notes:
- Some steps sequential, some parallel? No - typically all sequential.
- Combine steps into blocks? Yes.
- Has loops or iteration? Yes.
- Has “conditional”? Yes.
If a recipe is like a program, is a kitchen like a computer?

Let’s see if we can follow the analogy:

- Bowls (or pans) are like computer memory: they hold things during the activity.
- Bowls have names (kind of): big bowl, glass bowl, etc.
- Memory has names: corna, cornb, side.
- You can change what is in a bowl by mixing something in.
- You can change a variable (memory location) by "mixing" something in.
- You can change the contents of a bowl by emptying and then refilling.
- You can change the contents of a variable location in memory by *overwriting*. It's like the empty and refill operation in one swoop.
- You can test the value of a bowl by looking and tasting.
- You can test the value of a variable location in memory by boolean operators, e.g., =, <,>, etc.
WALLPAPER

1. input corna, cornb
2. input side
3. for i <- 1 to 100
   1. for j <- 1 to 100
      1. x <- corna + i * side/100
      2. y <- cornb + j * side/100
      3. c <- int( x*x + y*y )
      4. if c is even then plot( i,j )

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>corna</td>
<td>value</td>
</tr>
<tr>
<td>cornb</td>
<td>value</td>
</tr>
<tr>
<td>side</td>
<td>value</td>
</tr>
<tr>
<td>i</td>
<td>value</td>
</tr>
<tr>
<td>j</td>
<td>value</td>
</tr>
<tr>
<td>x</td>
<td>value</td>
</tr>
<tr>
<td>y</td>
<td>value</td>
</tr>
<tr>
<td>c</td>
<td>value</td>
</tr>
</tbody>
</table>
Let’s “run” the program

WALLPAPER

1. input corna, cornb
2. input side
3. for i <- 1 to 100
   1. for j <- 1 to 100
      1. x <- corna + i * side/100
      2. y <- cornb + j * side/100
      3. c <- int( x*x + y*y )
      4. if c is even then plot( i,j )
Result of running program
Is this a computer language?

WALLPAPER

1. input corna, cornb
2. input side
3. for i <- 1 to 100
   1. for j <- 1 to 100
      1. x <- corna + i * side/100
      2. y <- cornb + j * side/100
      3. c <- int( x*x + y*y )
      4. if c is even then plot( i,j )

No, it is called pseudo code.
WALLPAPER

1. input corna, cornb
2. input side
3. for i <- 1 to 100
   4. for j <- 1 to 100
      5. x <- corna + i * side/100
      6. y <- cornb + j * side/100
      7. c <- int( x*x + y*y )
      8. if c is even then plot( i,j )

Java equivalent

```java
1.a corna = readCorn();
1.b cornb = readCorn();
2. side = readSide();
3.a for(int i=1; i<=100; i++){
4.a for(int j=1; j<=100; j++){
5.   x = corna + i * side/100;
6.   y = cornb + j * side/100;
7.   c = (int)(x*x + y*y);
8.a  if( c%2 == 0 ){  
8.b   graph.putPixel(i,j,1);
8.c  } //end if
4.b } //end for
3.b } //end for
```
Have to give “type” to each variable

float corna;
float cornb;
float side;
float x;
float y;
int c;

corna = readCorn();
cornb = readCorn();
side = readSide();
for(int i=1; i<=100; i++){
    for(int j=1; j<=100; j++){
        x = corna + i * side/100;
        y = cornb + j * side/100;
        c = (int)(x*x + y*y);
        if( c%2 == 0 ){
            graph.putPixel(i,j,1);
        }
    }
}

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>corna</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cornb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>side</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type choices (for now): {int, float, boolean}
Have to “wrap” code in method and class

```java
public class Wallpaper {

    public static void main(String [] args) {
        float corna;
        float cornb;
        float side;
        float x;
        float y;
        int c;
        corna = readCorn();
        cornb = readCorn();
        side = readSide();
        for(int i=1; i<=100; i++) {
            for(int j=1; j<=100; j++) {
                x = corna + i * side/100;
                y = cornb + j * side/100;
                c = (int)(x*x + y*y);
                if( c%2 == 0 ) {
                    graph.putPixel(i, j, 1);
                } // if
            } // for
        } // for
    } // main
} // Wallpaper
```
public class Wallpaper {

    public static void main(String[] args) {
        Graph graph = new Graph();
        float corna;
        float cornb;
        float side;
        float x;
        float y;
        int c;
        corna = readCorn();
        cornb = readCorn();
        side = readSide();
        for (int i = 1; i <= 100; i++) {
            for (int j = 1; j <= 100; j++) {
                x = corna + i * side / 100;
                y = cornb + j * side / 100;
                c = (int) (x * x + y * y);
                if (c % 2 == 0) {
                    graph.putPixel(i, j, 1);
                } //if
            } //for
        } //for
    } //main
} //Wallpaper

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>corna</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>cornb</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>side</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>graph</td>
<td>Graph</td>
<td></td>
</tr>
</tbody>
</table>
public class Wallpaper {

    public static void main(String[] args){
        Graph graph = new Graph();
        [...]
        corna = readCorn();
        cornb = readCorn();
        side = readSide();
        [...]
            graph.putPixel(i,j,1);
        [...]
    } // main

    public float readCorn(){...} // I will supply this method
    public float readSide(){...} // I will supply this method
    // The method putPixel is supplied in the Graph class

} // Wallpaper
Now we have to compile the code

```java
public class Wallpaper {

    public static void main(String [] args){
        Graph graph = new Graph();
        [...]
        corna = readCorn();
        cornb = readCorn();
        side = readSide();
        [...]
        graph.putPixel(i,j,1);
        [...]
    } //main

    public float readCorn(){...} //I will supply this method
    public float readSide(){...} //I will supply this method
    //The method putPixel is supplied in the Graph class

} //Wallpaper
```

Now are we done?
Now we have to run the code

```java
public class Wallpaper {

    public static void main(String[] args) {
        Graph graph = new Graph();
        [...]
        corna = readCorn();
        cornb = readCorn();
        side = readSide();
        [...]
        graph.putPixel(i, j, 1);
        [...]
    }

    public static float readCorn() {...} //I will supply this method
    public static float readSide() {...} //I will supply this method
    //The method putPixel is supplied in the Graph class

} //Wallpaper
```

Now are we done?
Yes, but let's try for real.
Opening file in JCreator

Note I need two files in same folder: Wallpaper and Graph.