Main Class named as Ass0:

```
import java.util.*;
import java.io.*;

class Ass0 {
    public static void main(String[] args) throws Exception {
        FileWriter flo = new FileWriter("output.txt");// creating outputfile and
object to write in it
        Scanner sc = new Scanner(new File("input.txt"));// object to read input
file
        int epochs = 100; // number of observations to take average
        String flow = sc.hasNext() ? sc.nextInt() + " " + sc.nextInt() + "\n" :
"";// reading input
        while (sc.hasNext()) {// checking if data is available in file
            float pg = sc.nextFloat();// reading probability from file
            int steps = 0, pc = 0, width = sc.nextInt();// reading width from
file
            Walker wk = new Walker();// creating object for infiltrator
            for (int epoch = 0; epoch < epochs; epoch++) {// simulation starts
fpr 100 epochs
                String[] result = wk.start((1 - pg), width).split(" ");//
spliting the strimg to get time and success
                steps += Integer.parseInt(result[0]);
                pc += Integer.parseInt(result[1]);
            }
            flow += pg + " " + width + " " + (steps / epochs) + " " + (pc * 1.0 /
epochs) + "\n";
            // given probability and width -> average time taken probability of
success in
            // crossing border
        }
        flo.write(flow);
        flo.close();
        sc.close();

    }
}
```

Clock Class:

```
class Clock {
    private int t = 0; // recording time

    int next() {
        t += 10;// increse time by 10sec
        return t;
    }

    int nextinf() {
        t += t * 100;// increse time by its 100 times say infinity when
infiltrator stucks
        return t;
    }

    void reset() {
        t = 0;// resetting clock
    }

    int getTime() {
        return t;// sending time
    }
}
```

Wall class to Simulate Border:

```
public class Wall {
    Boolean w1 = false;// sensor1
    Boolean w2 = false;// sensor2
    Boolean w3 = false;// sensor3
    Boolean current = false;// sensor on which infiltrator is standing

    // situation faced by infiltrator
    // | |#cr| |
    // |#s1|#s2|#s3|

    void wallRender(float prob) {
        w1 = prob > Math.random();// weighted probability
        w2 = prob > Math.random();// weighted probability
        w3 = prob > Math.random();// weighted probability
        current = prob > Math.random();// weighted probability
    }
}
```

Walker Class to simulate infiltrator :

```
public class Walker {// class to simulate the infiltrator
    Wall wall = new Wall();// object to simulate border
    Clock clk = new Clock();// object to simulate time

    int checkMove(float prob) {// method to check if move is possible
        wall.wallRender(prob);// changing configurations of border sensor
        clk.next();// time increment
        if (wall.current && (wall.w1 || wall.w2 || wall.w3)) {// condition for
infiltrator to move
            return 1;
        }
        return 0;
    }

    String start(float prob, int width) {// starting infiltrator to cross border
        int pc = 0;
        int i, moves = 0;
        clk.reset();// resetting clock
        for (i = 0; i < 4000 && moves < width; i++) {// looping infiltrator to
move
            moves += checkMove(prob);
        }
        if (i < 4000) {// if infiltrator successfully crossed the border
            pc++;
        } else {
            clk.nextinf();// if infiltrator was unable to cross border in
40_000sec(666min(11hrs)) then
                          // time of crossing is assumed infinity
        }
        return clk.getTime() + " " + pc;// returning time and success
    }
}
```

Class to create input file for program:

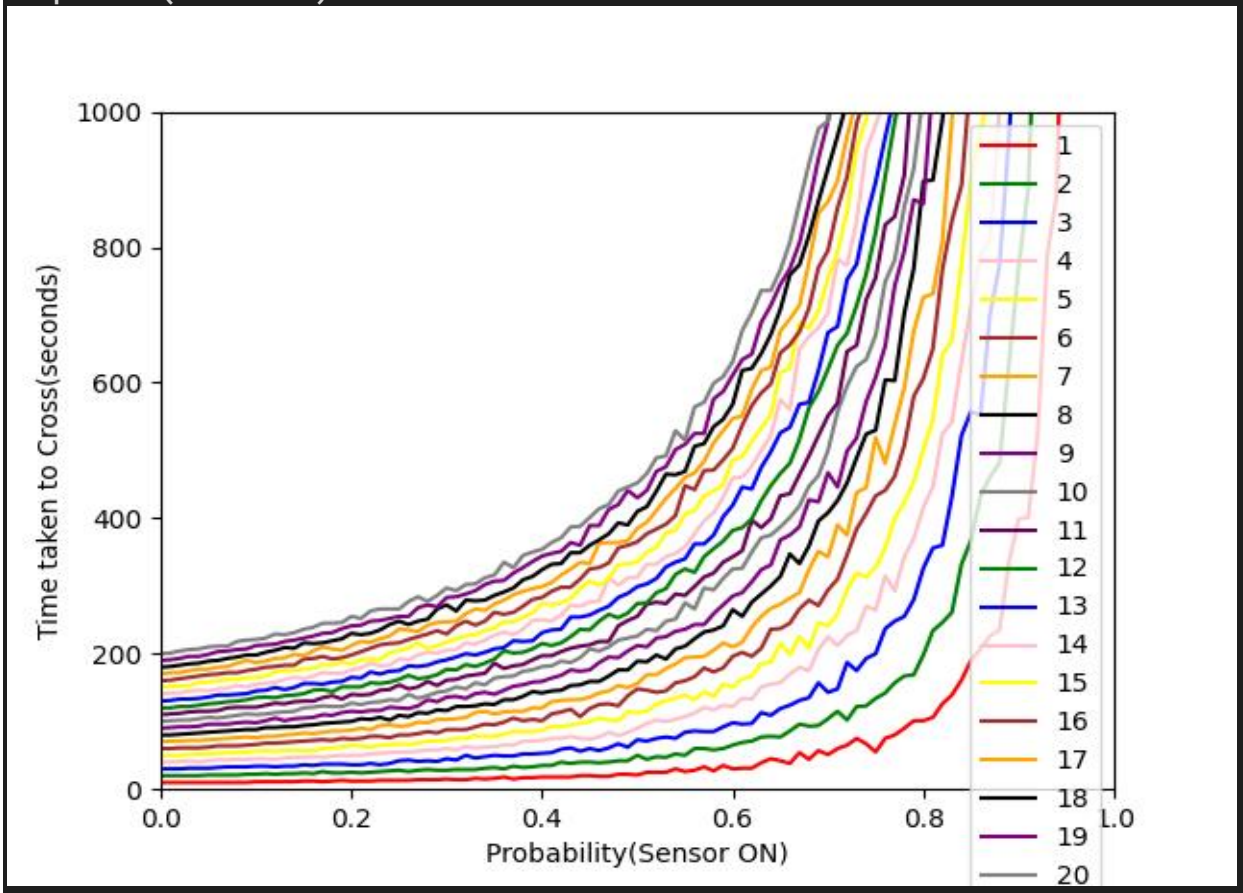
```
class CreateInpFile {
    public static void main(String[] args) throws Exception {
        FileWriter fw = new FileWriter(new File("input.txt"));// Writing an Input
File Sensor( prob | width );
        Scanner sc = new Scanner(System.in); // Creacting scanner object to take
input from console
        System.out.println("want to continue with default input(y/N)"); //
default is width(0-20)
        String inp = sc.next(); // reading input
        float minp = 0, maxp = 1;// default values for probability ranging from 0
to 1
        int minw = 1, maxw = 20;// default values for width ranging from 0 to 20
        if (inp.equalsIgnoreCase("N")) {// checking input
            System.out.println("Import range (min-width(>=1) and max-
width(<10000))");// getting custom input
            minw = sc.nextInt();
            maxw = sc.nextInt();
            minw = minw >= 1 ? minw : 1;
            maxw = maxw >= minw ? maxw : 20;// min with <max width
        }

        String flw = minw + " " + maxw + "\n"; // string containing range
        for (float i = minp; i <= maxp; i += 0.01) {
            for (int j = minw; j <= maxw; j += 1) {
                flw += i + " " + j + "\n";// all posibilities corresponding to
with are added to string

            }
        }
        sc.close();// scanner object is closed
        fw.write(flw);// string is written to file
        fw.close();// fileclosed
    }
}
```

Python program to plot graphs:

```
import matplotlib.pyplot as plt
import numpy as np

# List of colors
clr=["red","green","blue","pink","yellow","brown","orange","black","purple","grey
","#670056","green","blue","pink","yellow","brown","orange","black","purple","gre
y","#670056"]
# range of x and y axis on graph
plt.axis([0,1,0,1000])
#opening output file
fl=open("output.txt")
#spliting fist line to get range of width of borde
rng=fl.readline().split()
#dictionaries to store data from input file
waxs,waxp={},{}
# data to keep on x axis 0 0.01 0.02 ..... 0.99 1.00
xaxs=list(map(lambda i:i/100,[i for i in range(0,101,1)]))
# itterating from min-with to max-width
for i in range(int(rng[0]),int(rng[1])):
    #storing time value for each with from prob 0 to 1
    yaxs=[]
    rd=True # making while condition true initially
    while rd:
        rd=fl.readline()#reading each line of output file
        arr=rd.split(" ")#splitting the line-> prob given | width | time taken|
success probability to cross
        if len(arr)==4:#storing time corresponding to given probability
            if arr[1] in waxs.keys():
                waxs[arr[1]].append(round(float(arr[2])))
                waxp[arr[1]].append(float(arr[3]))
            else:
                waxs[arr[1]]=[round(float(arr[2]))]
                waxp[arr[1]]=[float(arr[3])]
            #xaxs.append(int(arr[2]))
            # p[arr[0]]=int(arr[2])
            # w[arr[1]]=arr[3]
fl.close()
for i in waxs.keys():#plotting each case(width range) on same graph of time taken
    if(int(rng[0])-int(rng[1])<21):
        plt.plot(xaxs,waxs[i],color=clr[int(i)-int(rng[0])],label=i)
    else:
        plt.plot(xaxs,waxs[i],label=i)
plt.xlabel("Probability(Sensor ON)")
```

```
plt.ylabel("Time taken to Cross(seconds)")
plt.legend()
plt.savefig("3blk.png")#saving graph
plt.show()#showing graph
plt.axis([0.85,1,0,1])

for i in waxp.keys():#plotting each case(width range) of probability of success
    if(int(rng[0])-int(rng[1])<21):
        plt.plot(xaxs,waxp[i],color=clr[int(i)-int(rng[0])-1],label=i)
    else:
        plt.plot(xaxs,waxp[i],label=i)

plt.xlabel("Probability(Sensor ON)")
plt.ylabel("Probability(Crossing Border)")
plt.legend()
plt.savefig("3blkp.png")
plt.show()
```

Graph of P(Sensor ON) vs Time Taken:



P(sensor ON) vs P(Crossing Border)