Java Projects

Professor William J. Wolfe
william.wolfe@angelo.edu
Angelo State University

Description: This document contains several Java projects, progressing from simple to difficult. All the projects were developed using the (free) Netbeans IDE and the (free) Java JDK. All the code is included, and users are encouraged to take each project (especially the later ones) and convert them into more elaborate applications of their own design.

Project Index

Java Project: GettingStarted1

Description: Some simple java exercises for getting accustomed to the Netbeans interface and simple java code.

Create a new Netbeans Project, name it **GettingStarted1**

Netbeans > file > new project > choose project: java application <next>
Project Name: GettingStarted <finish>

Your screen will now look something like:

```
//////////////////////////////////////////////////////////
/*
 * To change this license header, Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gettingstarted1;

/**
 *
 * @author wwolfe3
 */
public class GettingStarted1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //Your java code goes here
    }

}
//////////////////////////////////////////////////////////
```

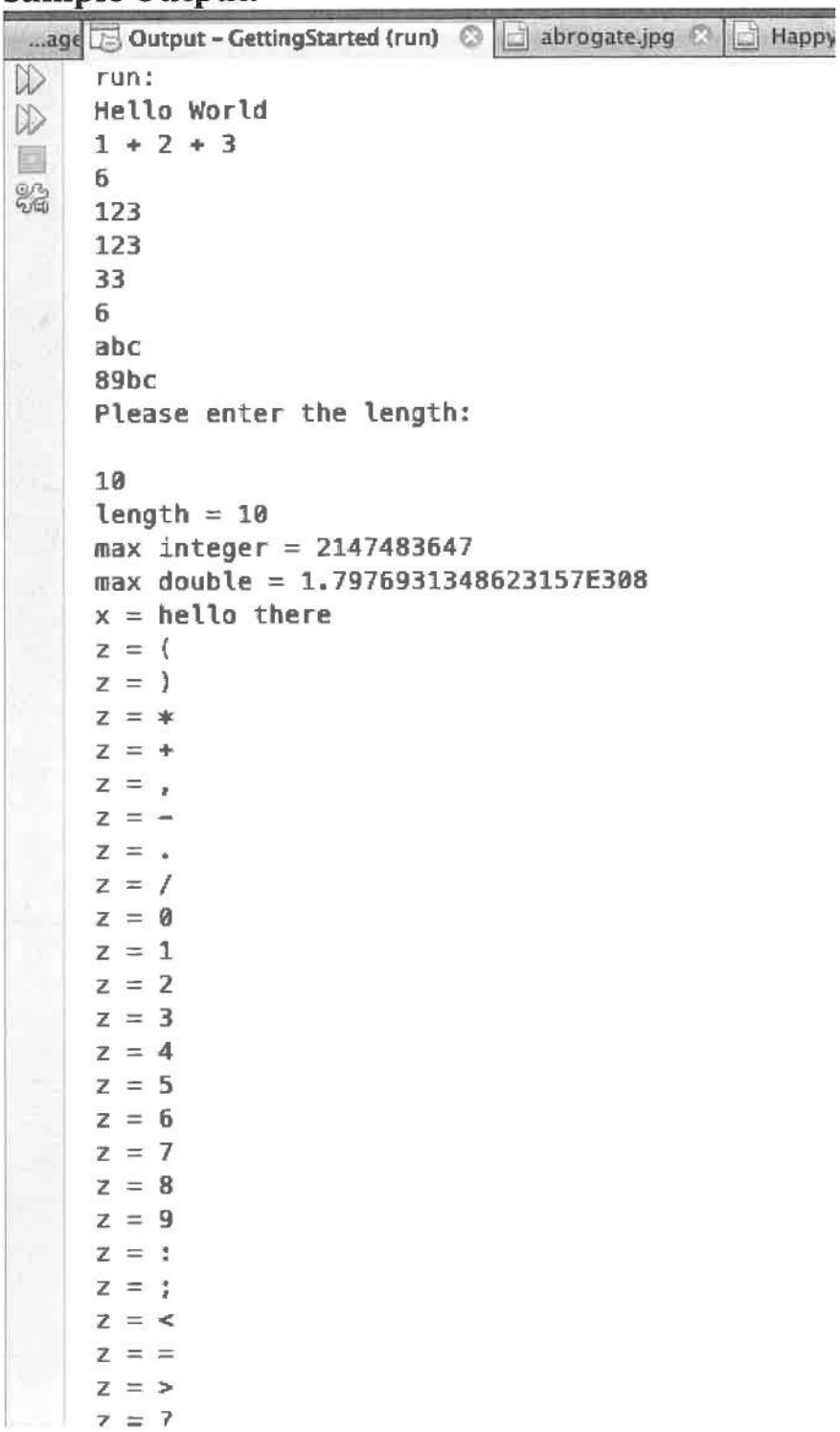
This code will be delted and replaced with the code on the following page.

Delete all the above code, replace it with the follow code:

```
package gettingstarted1;
import java.util.Scanner;


public class GettingStarted1 {

    public static void main(String[] args) {
        System.out.println("Hello World");

        System.out.println("1 + 2 + 3");
        System.out.println(1 + 2 + 3);
        System.out.println("1" + 2 + 3);
        System.out.println(1 + "2" + 3);
        System.out.println(1 + 2 + "3");
        System.out.println(1 + 2 + 3);

        System.out.println("a" + "b" + "c");

        int a = 89;
        System.out.println(a + "b" + "c");

        Scanner input = new Scanner(System.in);
        System.out.println("Please enter the length:");
        int length = input.nextInt();
        System.out.println("length = " + length);
        System.out.println("max integer = " + Integer.MAX_VALUE);
        System.out.println("max double = " + Double.MAX_VALUE);

        String x = "hello";
        x = x + " there";
        System.out.println("x = " + x);

        char z = 10036;
        for (int i = 40; i < 128; i++) {
            System.out.println("z = " + (char) i);
        }

    }

}
```

After clicking the green "Run" arrow on the menu bar you should get an output that looks like the follow (the computer will prompt you to enter a length → try 10)

Sample Output:



Java Project: GettingStarted2

Description: Some simple exercises to get started in Java programming.

Create a new Netbeans Project, name it **GettingStarted2**

Netbeans > new > project > java application > GettingStarted2

Type in code:

```
//////////////////////////////////////////
package gettingstarted2;

public class GettingStarted2 {

    public static void main(String[] args) {

        //A = pi r squared
        System.out.println("area = " + 3.14 * 5.0 * 5.0);
        double r = 5.0;
        double area = Math.PI * r * r;
        System.out.println("area = " + area);
        System.out.println("max integer = " + Integer.MAX_VALUE);
        System.out.println("max double = " + Double.MAX_VALUE);
    }

}
```

When it's typed in, click the "Run" arrow.

Sample Output:

```
area = 78.5
area = 78.53981633974483
max integer = 2147483647
max double = 1.79769313486231570E308
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Project: GettingStarted3

Description: GettingStarted3: This program prompts the user for a score (0 -100) and then presents the associated letter grade.

Create a new Netbeans Project, name it **GettingStarted3**

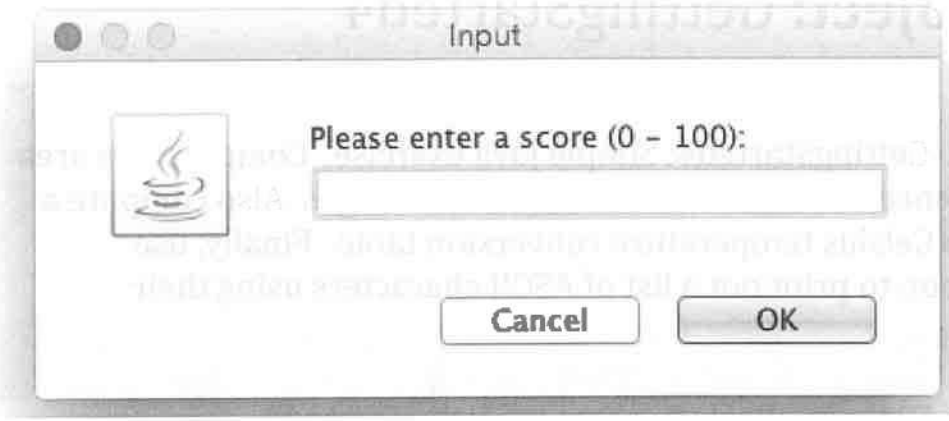
Netbeans > new > project > java application > GettingStarted3

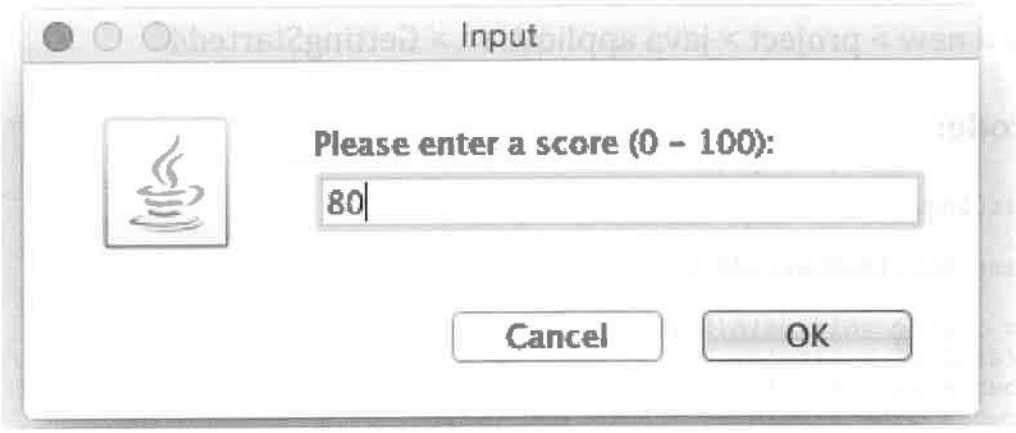
Type in code:

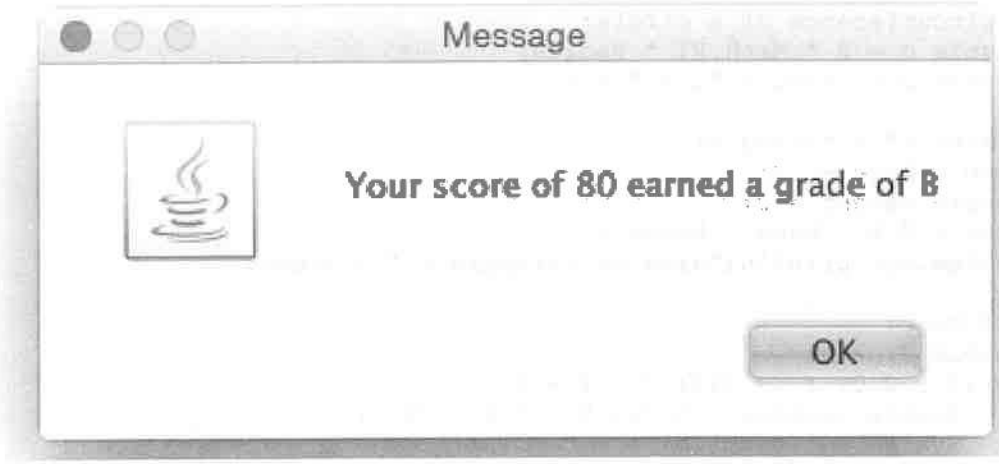
```
package gettingstarted3;

import javax.swing.JOptionPane;

public class GettingStarted3 {

    public static void main(String[] args) {
        //yesNoExample();
        //dialogWithIf();
        //dialogCircleArea();
        convertScoreToLetter();
    }

    public static void yesNoExample() {
        JOptionPane.showMessageDialog(null, "OK to continue",
"Warning", JOptionPane.WARNING_MESSAGE);

    }

    private static void dialogWithIf() {
        String input = JOptionPane.showInputDialog("Type
Something");
        System.out.println("Input = " + input);
        int choice = JOptionPane.showConfirmDialog(null, "Make
your choice");
        System.out.println("choice = " + choice);
        if (choice == 0) {
            System.out.println("You chose \"Yes\" ");
        }
        if (choice == 1) {
            System.out.println("You chose \"No\" ");
        }
        if (choice == 2) {
            System.out.println("You chose \"Cancel\" ");
```

```
        }
    }

    private static void dialogCircleArea() {
        String radius = JOptionPane.showInputDialog("Please enter
the circle radius:");
        double r = Double.parseDouble(radius);
        System.out.println("You entered a radius of: " + r);
        System.out.println("Area = " + Math.PI * Math.pow(r, 2));
    }

    private static void convertScoreToLetter() {
        int score =
Integer.parseInt(JOptionPane.showInputDialog("Please enter a
score (0 - 100):"));
        char letter;
        if (score >= 90 ) {
           letter = 'A';
        } else if ( score >= 80 ) {
           letter = 'B';
        } else if ( score >= 70 ) {
           letter = 'C';
        } else if ( score >= 60 ) {
           letter = 'D';
        } else {
           letter = 'F';
        }
        //System.out.println("Your score of " + score + " earned
a grade of " + letter);
        JOptionPane.showMessageDialog(null, "Your score of " +
score + " earned a grade of " + letter);


    }

}
```

Sample Output:


Input
Please enter a score (0 - 100):
Cancel
OK


Input
Please enter a score (0 - 100):
80
Cancel
OK


Message
Your score of 80 earned a grade of B
OK

Java Project: GettingStarted4

Description: GettingStarted4: Simple Java exercise. Compute the area and cirumference of a circle and the area of a triangle. Also compute a Fahrenheit to Celsius temperature conversion table. Finally, use another for loop to print out a list of ASCII characters using their decimal code.

Create a new Netbeans Project, name it GettingStarted4

Netbeans > new > project > java application > GettingStarted4

Type in code:

```
package gettingstarted4;

public class GettingStarted4 {

    public static void main(String[] args) {
        //area of a circle:
        double radius = 10;
        double area = Math.PI * Math.pow(radius, 2);
        System.out.println("area = " + area);

        //circumference of a circle:
        double c = 2 * Math.PI * radius;
        System.out.println("c = " + c);

        //area of a triangle:
        double base = 10;
        double height = 3;
        area = 0.5 * base * height;
        System.out.println("area of triangle = " + area);

        //f to c:
        double f;
        for(f = 0.0; f <= 212; f = f + 1) {
            double celsius = 5.0/9.0 * ( f - 32 );
            System.out.println("f = " + (int)f + " c = " +
(int)celsius);
        }

        for(int i = 4000; i < 4128; i++) {
            System.out.println((char)i);
        }

    }
```

```
}
```

Sample output: (the output is very long, so here it is abreviated):

```
area = 314.1592653589793
c = 62.83185307179586
area of triangle = 15.0
f = 0 c = -17
f = 1 c = -17
f = 2 c = -16
f = 3 c = -16
f = 4 c = -15
f = 5 c = -15
f = 6 c = -14
f = 7 c = -13
f = 8 c = -13
f = 9 c = -12
f = 10 c = -12
f = 11 c = -11
f = 12 c = -11
f = 13 c = -10
f = 14 c = -10
f = 15 c = -9
f = 16 c = -8
f = 17 c = -8
.....
f = 204 c = 95
f = 205 c = 96
f = 206 c = 96
f = 207 c = 97
f = 208 c = 97
f = 209 c = 98
f = 210 c = 98
f = 211 c = 99
f = 212 c = 100
```

```
BUILD SUCCESSFUL (total time: 0 seconds)
```

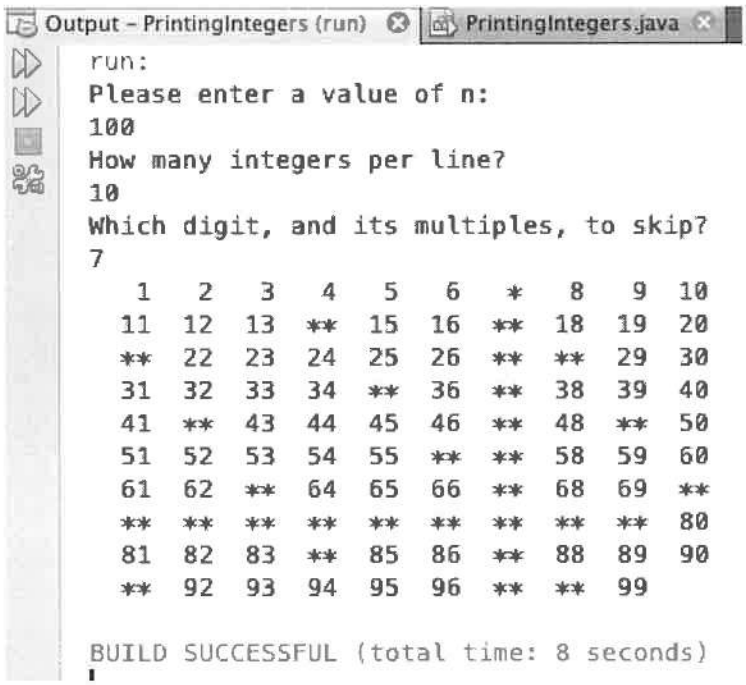
Java Project: PrintingIntegers

Create a new Netbeans Project, name it PrintingIntegers

Netbeans > file > new project > project > java application <next> > Project Name: PrintingIntegers > <finish>

Description: Print a list of integers in a rectangular layout, skipping those integers that contain, or are a multiple of, a specified digit. Uses a method (indexOf) to detect if a character is in a given string.

Sample Output:



```
package printingintegers;

import java.util.Scanner;

public class PrintingIntegers {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Please enter a value of n:");
        int n = input.nextInt();
        String nString = "" + n; //length of n treated as string
        int maxDigits = nString.length();
        System.out.println("How many integers per line?");
        int p = input.nextInt();
        System.out.println("Which digit, and its multiples, to
skip?");
        int m = input.nextInt();
        for(int i = 1; i < n; i++) {
            if (  ((i%m) == 0 ) || hasM(i,m) ){
                int iLength = ("" + i).length();
                String stars = getStars(iLength);
                System.out.format("%" + (maxDigits + 1) + "s",
stars);
            } else {
                System.out.format("%" + (maxDigits + 1) + "d", i);
            }
            if( i % p  == 0) {
                System.out.println("");
            }
        }
        System.out.println("");
        System.out.println("");
    }

    public static String getStars(int p) {
        String stars = "";
        for(int k = 0; k < p; k++) {
           stars += "*";
        }
        return stars;
    }

    public static boolean hasM(int i, int m) {
        char c = String.format("%s", m).charAt(0);
        String x = "" + i;
        return x.indexOf(c) >= 0;
    }

}
```

Java Project: Population

Description: Predicts the population based on assumptions about birth and death rate.

Sample Output:

run:
pop year 0 = 312032486
pop year 1 = 314812582
pop year 2 = 317592678
pop year 3 = 320372774
pop year 4 = 323152870
pop year 5 = 325932966

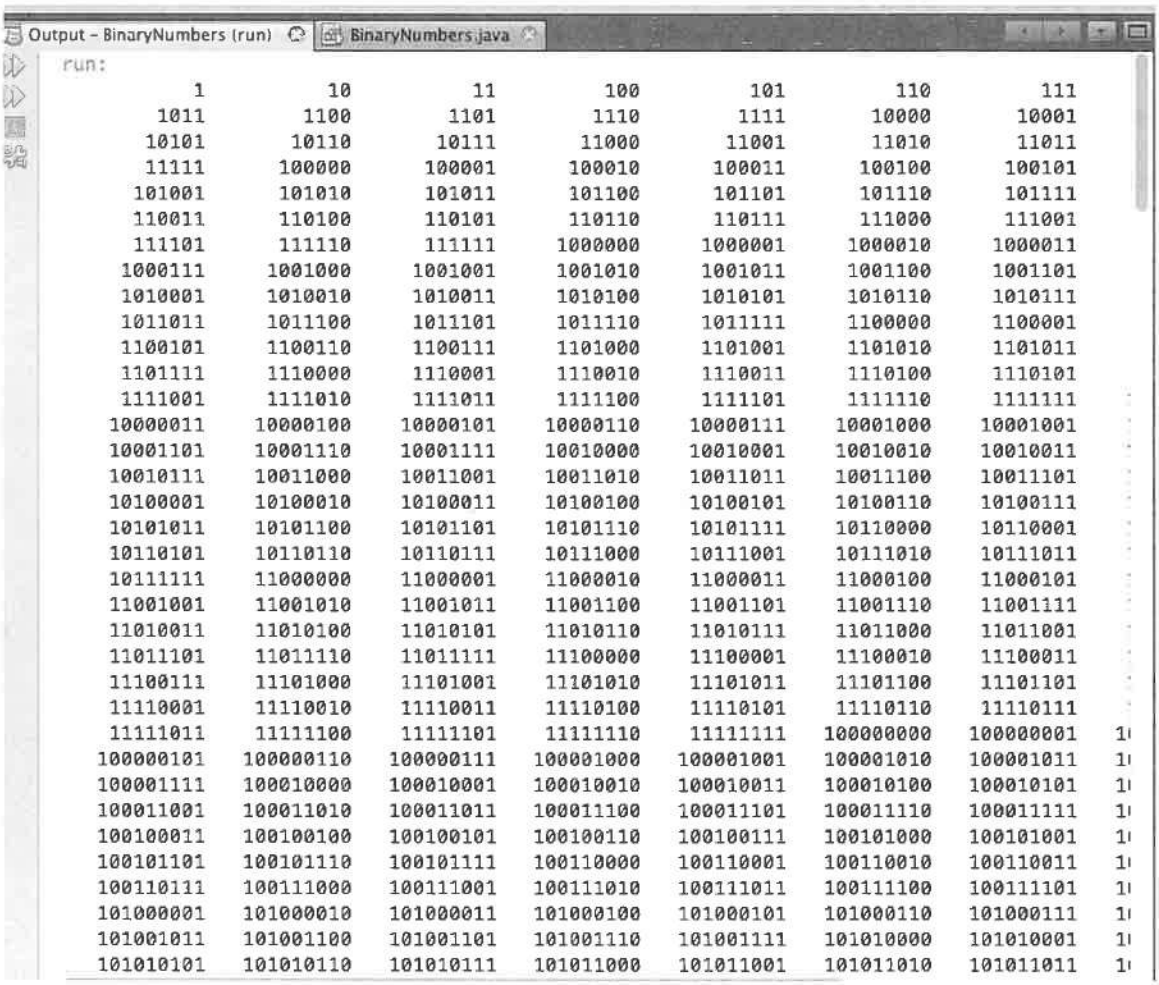
```
package population;

public class Population {

    public static void main(String[] args) {
        //Given: current pop, birth rate, death rate, immig. rate
        //predict the population for 5 years
        //a birth every 7 seconds.
        //a death every 13 seconds.

        //an immigrant every 45 seconds.
        int pop = 312032486;
        System.out.println("pop year 0 = " + pop);
        int n = 365 * 24 * 60 * 60;
        int births = n / 7;
        int deaths = n / 13;
        int immigrants = n / 45;
        int net = births + immigrants - deaths;
        int years = 5;
        for (int i = 1; i <= years; i = i + 1) {
            pop = pop + net;
            System.out.println("pop year " + i + " = " + pop);
        }
    }

}
```

Java Project: BinaryNumbers

Description: Print a list of binary numbers, in order. It uses a method to convert a decimal number to its binary equivalent.

Netbeans > file > new project > project > java application <next> > Project Name: BinaryNumbers > <finish>

Sample Output:



```
package binarynumbers;

public class BinaryNumbers {

   public static void main(String[] args) {
        int n = 2015;
        String nBinary = convertToBinary(n);
        int maxDigits = nBinary.length();
        int p = 10; // numbers per line
        for (int i = 1; i <= n; i++) {
            System.out.format("%" + (maxDigits + 1) + "s",
convertToBinary(i));
            if ((i % p) == 0) {
                System.out.println("");
            }
        }
    }

    public static String convertToBinary(int k) {
        String binary = "";
        while (k > 0) {
            if (k % 2 == 1) {
                binary = 1 + binary;
            } else {
                binary = 0 + binary;
            }
            k = k / 2;
        }
        return binary;
    }
}
```

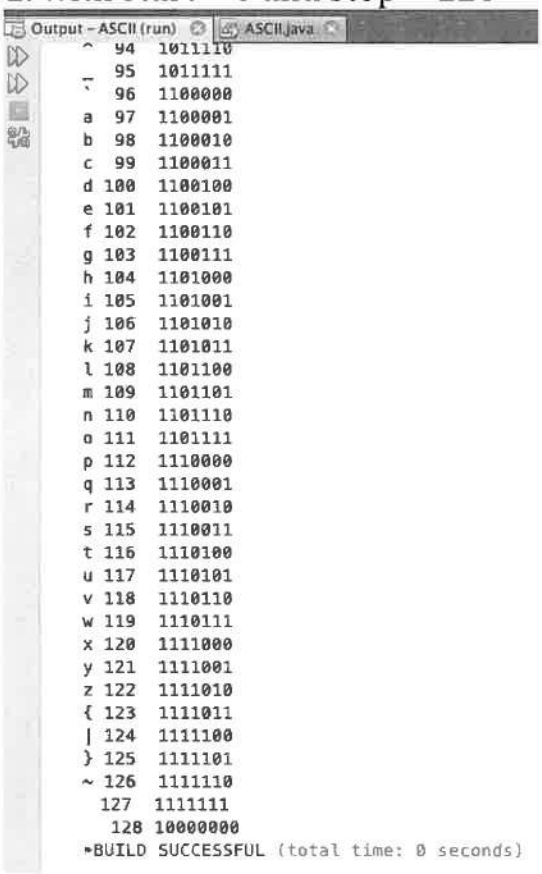
Java Project: ASCII

Description: Print a list of ASCII (or Unicode) characters with their decimal and binary representations.

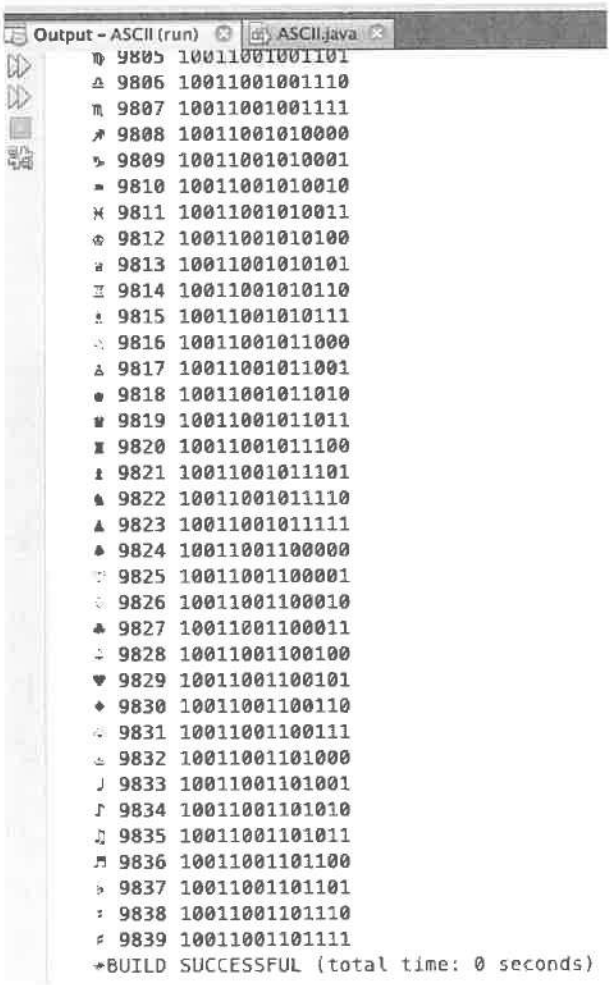
Netbeans > file > new project > project > java application <next> > Project Name: ASCII> <finish>

Sample Output:

1. with start = 0 and stop = 128



2. with: start = 9728 and stop = 9839;


Output - ASCII (run)
ASCII.java
9805 10011001001101
9806 10011001001110
9807 10011001001111
9808 10011001010000
9809 10011001010001
9810 10011001010010
9811 10011001010011
9812 10011001010100
9813 10011001010101
9814 10011001010110
9815 10011001010111
9816 10011001011000
9817 10011001011001
9818 10011001011010
9819 10011001011011
9820 10011001011100
9821 10011001011101
9822 10011001011110
9823 10011001011111
9824 10011001100000
9825 10011001100001
9826 10011001100010
9827 10011001100011
9828 10011001100100
9829 10011001100101
9830 10011001100110
9831 10011001100111
9832 10011001101000
9833 10011001101001
9834 10011001101010
9835 10011001101011
9836 10011001101100
9837 10011001101101
9838 10011001101110
9839 10011001101111
BUILD SUCCESSFUL (total time: 0 seconds)

```
package ascii;
public class ASCII {
    public static void main(String[] args) {
        //note: Greek upper case: 913 --> 937
        //  Greek lower case: 945 --> 969
        int start = 9728;
        int stop  = 9839;
        int maxDigits = ("" + stop).length() + 1;
        int maxBinaryDigits = convertToBinary(stop).length() + 1;
        for(int i = start; i <= stop; i++) {
            System.out.format("%3s", (char)i);
            System.out.format("%" + maxDigits + "s", i);
            System.out.format("%" + maxBinaryDigits + "s",
convertToBinary(i));
            System.out.println("");
        }
       System.out.format("%3s", (char)9728);
    }
    public static String convertToBinary(int k) {
        String binary = "";
        while (k > 0) {
            if (k % 2 == 1) {
                binary = 1 + binary;
            } else {
                binary = 0 + binary;
            }
            k = k / 2;
        }
        return binary;
    }
}
```

Java Project: Factorial

Description: Computes the factorial of a given integer in two ways: iterative (with a loop) and recursively (self-referencing method).

Sample Output:

run:
10 factorial (iterative) = 3628800
10 factorial (recursive) = 3628800
BUILD SUCCESSFUL (total time: 0 seconds)

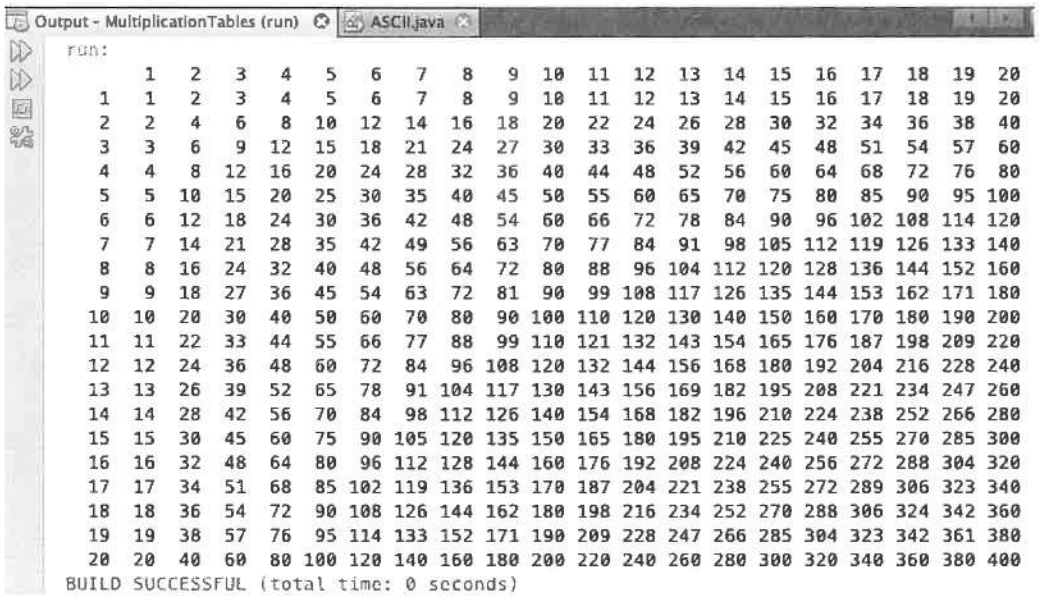
```
package factorial;

public class Factorial {

    public static void main(String[] args) {
        int n = 10;
        int factorial = 1;

        //first way (without recursion)
        for (int i = 1; i <= n; i++) {
            factorial = i * factorial;
        }

        System.out.println(n + " factorial (iterative) = " +
factorial);

        //second way (with recursion)
        factorial = getFactorial(n);

        System.out.println(n + " factorial (recursive) = " +
factorial);

    }

    public static int getFactorial(int n) {
        if (n == 0) {
            return 1;
        }
        return n * getFactorial(n - 1);
    }
}
```

Java Project: MultiplicationTables

Description: Print a multiplication table for a given integer size.

Netbeans > file > new project > project > java application <next> > Project Name: MultiplicationTables > <finish>

Sample Output: n = 20




```
     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 1   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 2   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40
 3   3   6   9  12  15  18  21  24  27  30  33  36  39  42  45  48  51  54  57  60
 4   4   8  12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76  80
 5   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95 100
 6   6  12  18  24  30  36  42  48  54  60  66  72  78  84  90  96 102 108 114 120
 7   7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112 119 126 133 140
 8   8  16  24  32  40  48  56  64  72  80  88  96 104 112 120 128 136 144 152 160
 9   9  18  27  36  45  54  63  72  81  90  99 108 117 126 135 144 153 162 171 180
10  10  20  30  40  50  60  70  80  90 100 110 120 130 140 150 160 170 180 190 200
11  11  22  33  44  55  66  77  88  99 110 121 132 143 154 165 176 187 198 209 220
12  12  24  36  48  60  72  84  96 108 120 132 144 156 168 180 192 204 216 228 240
13  13  26  39  52  65  78  91 104 117 130 143 156 169 182 195 208 221 234 247 260
14  14  28  42  56  70  84  98 112 126 140 154 168 182 196 210 224 238 252 266 280
15  15  30  45  60  75  90 105 120 135 150 165 180 195 210 225 240 255 270 285 300
16  16  32  48  64  80  96 112 128 144 160 176 192 208 224 240 256 272 288 304 320
17  17  34  51  68  85 102 119 136 153 170 187 204 221 238 255 272 289 306 323 340
18  18  36  54  72  90 108 126 144 162 180 198 216 234 252 270 288 306 324 342 360
19  19  38  57  76  95 114 133 152 171 190 209 228 247 266 285 304 323 342 361 380
20  20  40  60  80 100 120 140 160 180 200 220 240 260 280 300 320 340 360 380 400
```

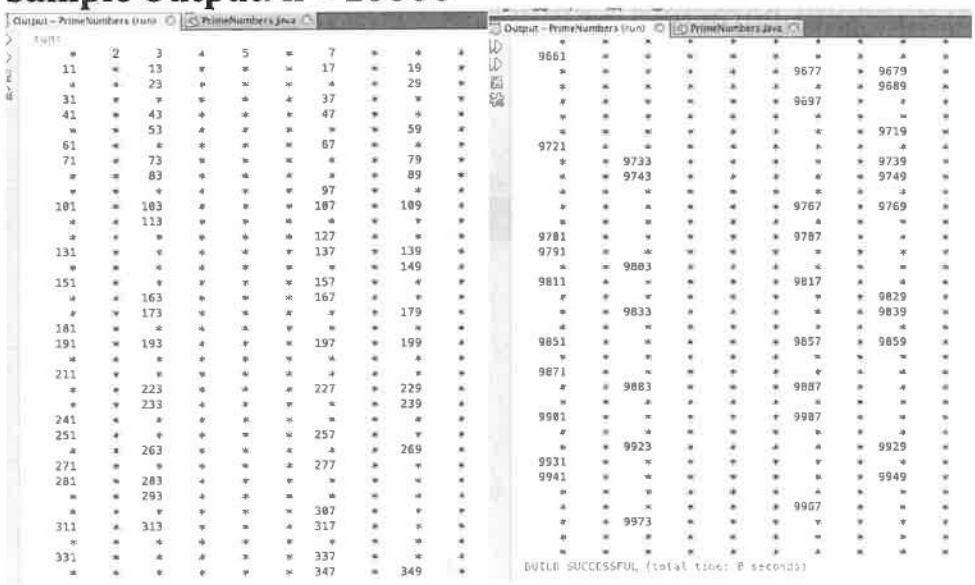
```
package multiplicationtables;
public class MultiplicationTables {

    public static void main(String[] args) {
        int n = 20;
        int maxDigits = ("" + n*n).length();
        System.out.format("%" + (maxDigits + 1) + "s", " ");
        for (int i = 1; i <= n; i++) {
            System.out.format("%" + (maxDigits + 1) + "d", i);
        }
        System.out.println("");
        for (int j = 1; j <= n; j++) {
            System.out.format("%" + (maxDigits + 1) + "d", j);
            for (int i = 1; i <= n; i++) {
               System.out.format("%" + (maxDigits + 1) + "d", i*j);
            }
            System.out.println("");
        }
    }
}
```

Java Project: PrimeNumbers

Description: Print out the prime numbers up to some maximum value.

Netbeans > file > new project > project > java application <next> > Project Name: **PrimeNumbers** > <finish>

Sample Output: n = 10000

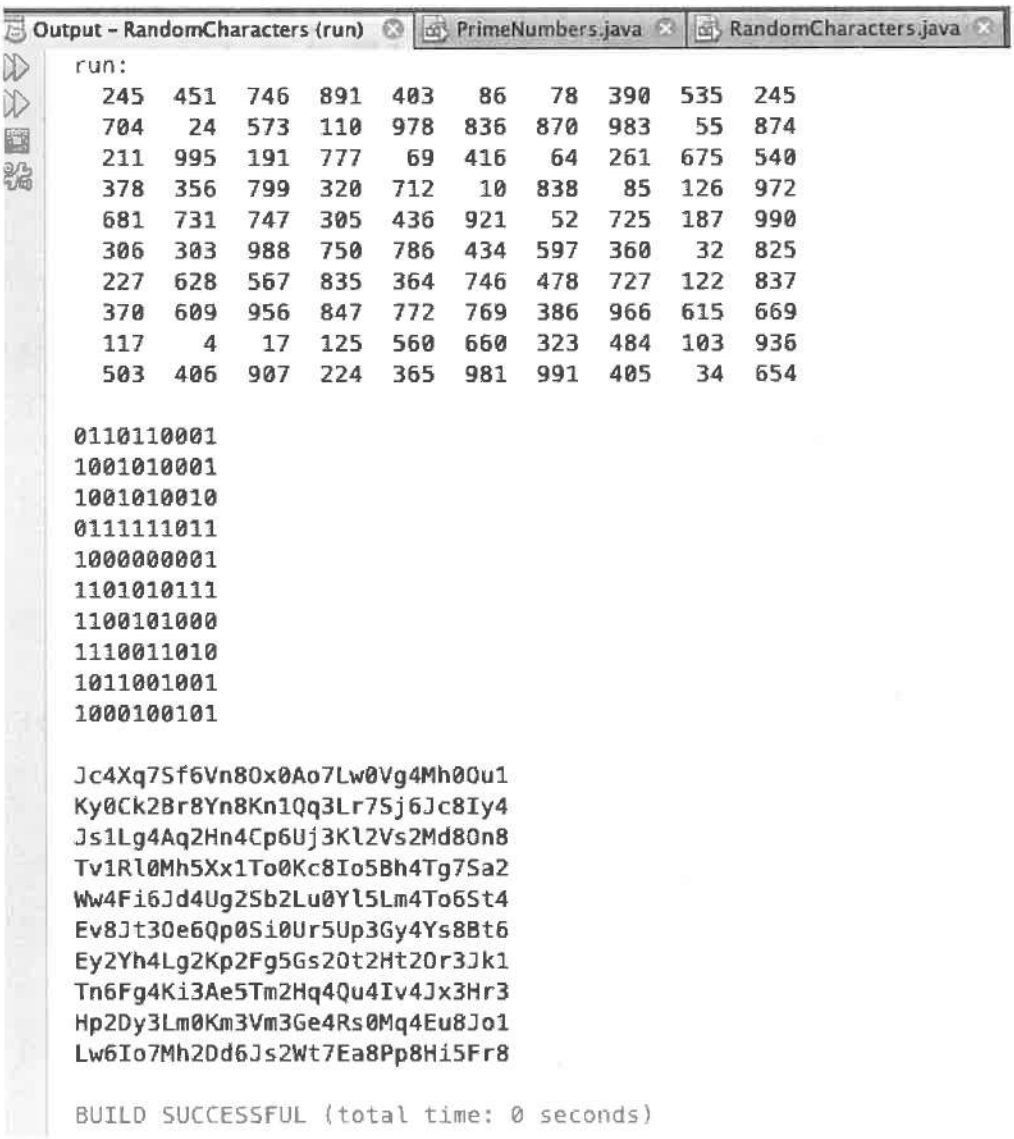


```
package primenumbers;
public class PrimeNumbers {
    public static void main(String[] args) {
        int n = 10000;
        int maxDigits = ("" + n).length();
        int p = 10;
        for (int i = 1; i <= n; i++) {
            if (isPrime(i)) {
                System.out.format("%" + (maxDigits + 1) + "d", i);
            } else {
                System.out.format("%" + (maxDigits + 1) + "s", "*");
            }
            if ( (i % p) == 0 ) System.out.println("");
        }
    }
public static boolean isPrime(int p) {
        boolean value = true;
        if (p == 1) return false;
        for (int i = 2; i <= p / 2; i++) {
            if (p % i == 0) {
                value = false;
            }
        }
        return value;
    }

}
```

Java Project: RandomCharacters

Description: Print random characters of all sorts.
Netbeans > file > new project > project > java application <next> > Project Name: RandomCharacters > <finish>

Sample Output:

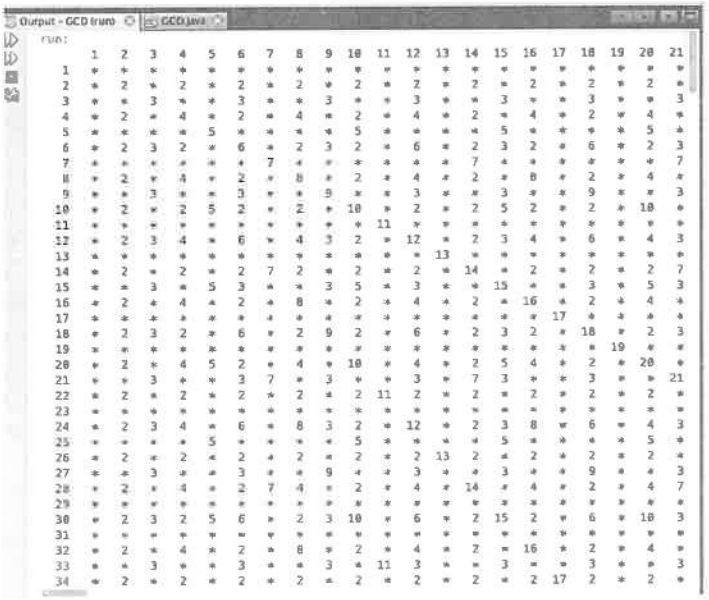


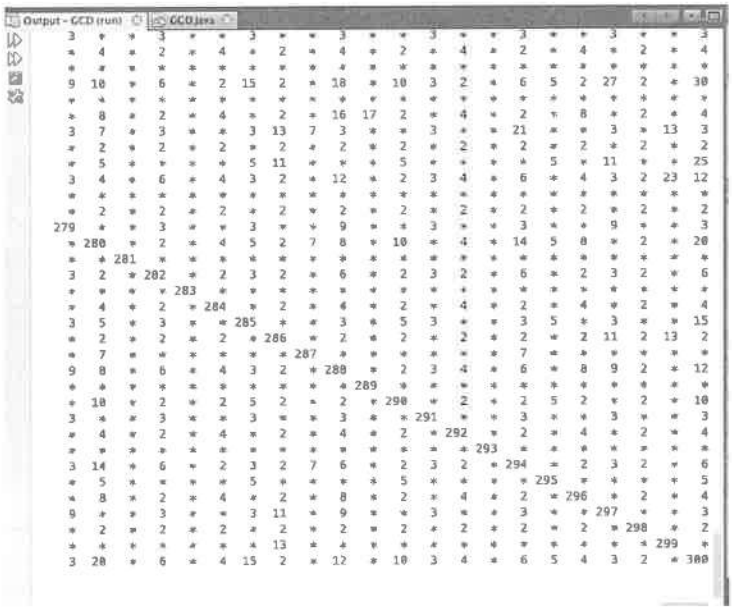
```
package randomcharacters;
import java.util.Random;
public class RandomCharacters {

    public static void main(String[] args) {
        Random r = new Random();
        int n = 10;
        int p = 1000;
        int maxDigits = ("" + p).length();
        //print an nxn block of random integers
        for (int j = 1; j <= n; j++) {
            for (int i = 1; i <= n; i++) {
                System.out.format("%" + (maxDigits + 1) + "d",
r.nextInt(p));
            }
            System.out.println("");
        }
        System.out.println("");

        int out;

        //print an nxn block of binary bits
        for (int j = 1; j <= n; j++) {
            for (int i = 1; i <= n; i++) {
                if (r.nextBoolean()) {
                    out = 1;
                } else {
                    out = 0;
                }
                System.out.format("%d", out);
            }
            System.out.println("");
        }
        System.out.println("");

        //print an nxn block of random letters and digits
        for (int j = 1; j <= n; j++) {
            for (int i = 1; i <= n; i++) {
                System.out.format("%c", getRandomChar(r, 'A', 'Z'));
                System.out.format("%c", getRandomChar(r, 'a', 'z'));
                System.out.format("%c", getRandomChar(r, '0', '9'));
            }
            System.out.println("");
        }
        System.out.println("");

    }

    public static char getRandomChar(Random r, char from, char to) {
        int k = r.nextInt((int)to - (int)from);
        char c = (char) (k + from);
        return c;
    }
}
```

Java Project: GCD

Description: Calculate and print the greatest common divisor of a pair of numbers.

Sample Output:

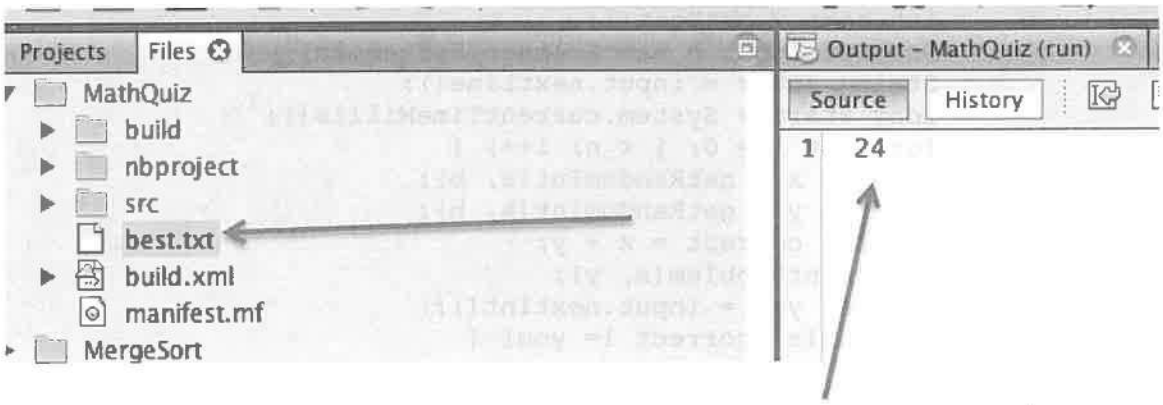




```
package gcd;
public class GCD {
    public static void main(String[] args) {
        int n = 300;
        int maxDigits = ("" + n).length();
        System.out.format("%" + (maxDigits + 1) + "s", " ");
        for (int i = 1; i <= n; i++) {
            System.out.format("%" + (maxDigits + 1) + "d", i);
        }
        System.out.println("");
        for (int j = 1; j <= n; j++) {
            System.out.format("%" + (maxDigits + 1) + "d", j);
            for (int i = 1; i <= n; i++) {
                int thisGCD = gcd(i,j);
                if(thisGCD == 1) {
                    System.out.format("%" + (maxDigits + 1) + "s",
"*");
                } else {
                    System.out.format("%" + (maxDigits + 1) + "d",
gcd(i, j));
                }
            }
            System.out.println("");
        }
    }
    public static int gcd(int a, int b) {
        if (a == 0) return b;
        if (b == 0) return a;
        if (a > b) return gcd(b, a % b);
        return gcd(a, b % a);
    }
}
```

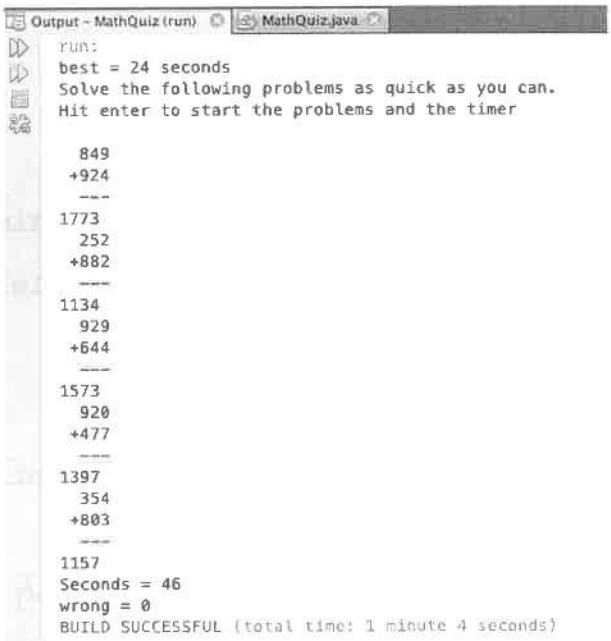
Java Project: MathQuiz

Description: Solve arithmetic problems as fast as you can.
Besides the jave file, the code uses a file called best.txt. There is only one number in best.txt, the fewest seconds that the user has taken to solve the randomly generated set of arithmetic problems.



best.txt has only one number in it, the best time (seconds) that the quiz has been completed in so far.

Sample Output:



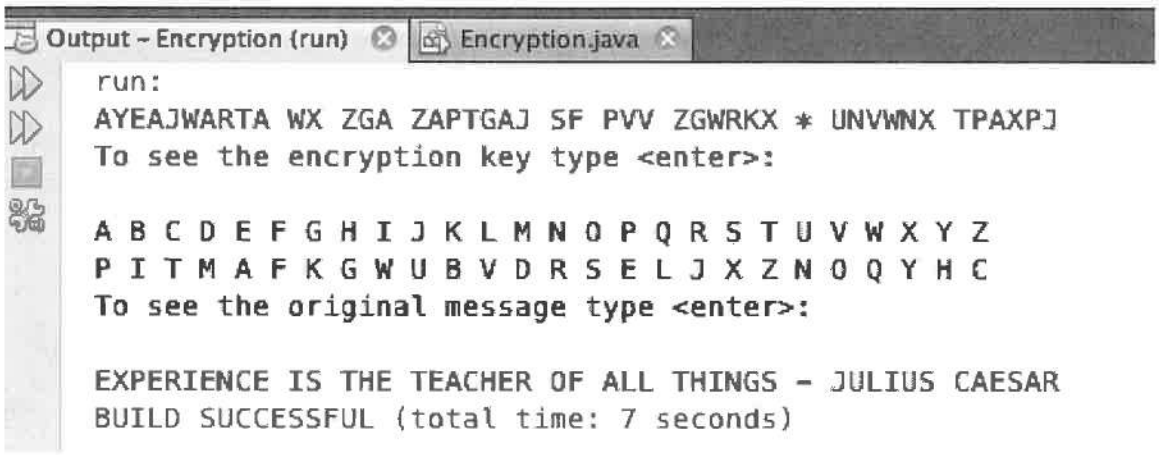
```
package mathquiz;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class MathQuiz {
    public static void main(String[] args) throws
FileNotFoundException {
        int n = 5;  //number of problems to solve
        int m = 3;  //number of digits in the numbers
        int a = (int) Math.pow(10, m - 1);
        int b = (int) Math.pow(10, m) - 1;
        int n_wrong = 0;
        int best = getBest();
        Scanner input = new Scanner(System.in);
        String enter = input.nextLine();
        long start = System.currentTimeMillis();
        for (int i = 0; i < n; i++) {
            int x = getRandomInt(a, b);
            int y = getRandomInt(a, b);
            int correct = x + y;
            printProblem(x, y);
            int you = input.nextInt();
            while (correct != you) {
                n_wrong++;
                System.out.println("incorrect, try again:");
                printProblem(x, y);
                you = input.nextInt();
            }
        }
        finish(start, best, n_wrong);
    }

    public static void finish(long start, int best, int n_wrong )
throws FileNotFoundException{
        long t = System.currentTimeMillis() - start;
        long yourTime = t / 1000;
        System.out.println("Seconds = " + yourTime);
        System.out.println("wrong = " + n_wrong);
        if (yourTime < best) {
            System.out.println("New best time: " + yourTime);
            File bestFile = new File("best.txt");
            PrintWriter output = new PrintWriter(bestFile);
            output.print(yourTime);
            output.close();
        }
    }

    public static int getBest() throws FileNotFoundException {
        File file = new File("best.txt");
        Scanner getBest = new Scanner(file);
        int best = getBest.nextInt();
        System.out.println("best = " + best + " seconds");
```

```
        System.out.println("Solve the following problems as quick as
you can.");
        System.out.println("Hit enter to start the problems and the
timer");
        return best;
    }

    public static void printProblem(int x, int y) {
        int length = ("" + x).length();
        System.out.println("  " + x);
        System.out.println(" +" + y);
        System.out.print("  ");
        for (int j = 0; j < length; j++) {
            System.out.print("-");
        }
        System.out.println("");
    }

    public static int getRandomInt(int a, int b) {
        return (int) ((b + 1 - a) * Math.random() + a);
    }
}
```

Java Project: Encryption

Description: Encode a message with an encryption cipher and then decode it. The cipher is a randomized, shuffled, alphabetic list.

Sample Output:




```
package encryption;
import java.util.Random;
import java.util.Scanner;
public class Encryption {

    public static void main(String[] args) {
        int n = 26;
        char[] alphabet = new char[n];
        char[] key = new char[n];
        for (int i = 0; i < n; i++) {
            alphabet[i] = (char) (i + 65);
            key[i] = alphabet[i];
        }
        key = createRandomKey(key);


        String input = "Experience is the teacher of all things -
Julius Caesar";
        input = input.toUpperCase();

        String output = encrypt(input, key);
        System.out.println(output);

        Scanner scanner = new Scanner(System.in);
        System.out.println("To see the encryption key type
<enter>:");
        String getEnter1 = scanner.nextLine();
```

```
        for (int i = 0; i < n; i++) {
            System.out.print(alphabet[i] + " ");
        }
        System.out.println("");
        for (int i = 0; i < n; i++) {
            System.out.print(key[i] + " ");
        }
        System.out.println("");
        System.out.println("To see the original message type
<enter>:");
        String getEnter2 = scanner.nextLine();
        System.out.println(input);
    }

    public static char[] createRandomKey(char[] key) {
        int n = key.length;
        int m = n * n;
        Random r = new Random();
        for (int p = 0; p < m; p++) {
            int swap1 = r.nextInt(n);
            int swap2 = r.nextInt(n);
            char temp = key[swap1];
            key[swap1] = key[swap2];
            key[swap2] = temp;

        }
        return key;
    }
    public static String encrypt(String input, char[] key) {
        String output = "";

        for (int k = 0; k < input.length(); k++) {
            char thisChar = input.charAt(k);
            if (thisChar == ' ') {
                output += ' ';
            } else if((int)thisChar > 91  || (int)thisChar < 65) {
                output += '*';
            } else {
                int index = (int) input.charAt(k) - 65;
                output += key[index];
            }

        }
        return output;
    }
}
```

Java Project: InsertionSort

Description: Sorts a random array of integers, using the method called "insertion sort".

Sample Output:
run:

Original array:
a[0] = 8
a[1] = 16
a[2] = 91
a[3] = 40
a[4] = 50
a[5] = 29
a[6] = 53
a[7] = 28
a[8] = 87
a[9] = 86

Sorted array:
a[0] = 8
a[1] = 16
a[2] = 28
a[3] = 29
a[4] = 40
a[5] = 50
a[6] = 53
a[7] = 86
a[8] = 87
a[9] = 91

BUILD SUCCESSFUL (total time: 0 seconds)

```
/*
 * Insertion sort -- ref: Algorithms Text, Cormen
 */
package insertionsort;

import java.util.Arrays;

public class InsertionSort {

    public static void main(String[] args) {
        //initialize an array of integers:
        final int N = 10;
        int[] array = new int[N];

        //generate ranom integers from 0 to 99:
        for (int i = 0; i < array.length; i++) {
            array[i] = (int)(100 * Math.random());
        }

        //print the random array:
        System.out.println("Original array:");
        printArray(array);

        insertionSort(array);
        //Java has a sort function: Arrays.sort(array);

        //print the sorted array
        System.out.println("Sorted array:");
        printArray(array);

    }

    private static void printArray(int[] a) {
        for(int i = 0; i < a.length; i++) {
            System.out.println("a[" + i + "] = " + a[i]);
        }

    }

    private static void insertionSort(int[] a) {
        for(int j = 1; j < a.length; j++) {
            int key = a[j];
            int i = j - 1;
            while( i >= 0  && a[i] > key) {
                a[i+1] = a[i];
                i = i - 1;
            }
            a[i+1] = key;
        }
    }

}
```

Java Project: CardGame

Description: This program "deals" a poker hand, and checks for pairs, three of kind, and four of a kind. Just the beginning of what could be a fund poker playing game.

Sample Output:

4♣ 4♦ 9♥ 9♣ T♥
Has a pair of 4's
BUILD SUCCESSFUL (total time: 0 seconds)

```
package cardgame;

import java.util.Scanner;

public class CardGame {

    public static int[] hand = new int[5];
    public static int[] deck = new int[52];

    public static void main(String[] args) {
        //lab4_exercise_1();
        lab4_exercise_4();
}

    private static int randomInteger(int a, int b) {
        return (int) ((b - a) * Math.random() + a);
    }

    private static void lab4_exercise_1() {
        //generate two integers
        //for timing:
        long start, stop;
        start = System.currentTimeMillis();
        System.out.println("start time = " + start);
        int n1, n2;
        int n_digits = 2;
        int a = (int) Math.pow(10, n_digits - 1);
        int b = (int) Math.pow(10, n_digits);
        n1 = randomInteger(a, b);
        n2 = randomInteger(a, b);
        System.out.println(" " + n1);
        System.out.println(" " + n2);
        System.out.println("+___");
        int correct = n1 + n2;
        Scanner input = new Scanner(System.in);
        int answer = input.nextInt();
```

```
        while (answer != correct) {
            System.out.println("Wrong, try again:");
            answer = input.nextInt();
        }
        System.out.println("Got it");
        stop = System.currentTimeMillis();
        System.out.println("It took you " + (stop - start) / 1000.0
+ " seconds.");

    }

    private static void lab4_exercise_4() {
        // choose a random card
        // model the deck as: 52 cards, indexed from 0 to 51.
        // print the whole deck:
        //printDeck();
        initializeDeck();
        shuffleDeck();
        dealPokerHand();
        sortHand();
        displayPokerHand();
        if (hasPair() >= 0) {
            System.out.println("Has a pair of " + getRank(hasPair()
% 13) + "'s");
        }

        if (hasThree() >= 0) {
            System.out.println("Has three " + getRank(hasThree() %
13) + "'s");
        }
      }

    private static int getSuit(int suit) {

        switch (suit) {
            case 0:
                return 9829; //hearts
            case 1:
                return 9830; //diamonds
            case 2:
                return 9824; //spades
            case 3:
                return 9827; //clubs
            default:
                return 9837; //flat
        }

    }

    private static char getRank(int rank) {
        if (rank > 0 && rank < 9) {
            return (char) (49 + rank);
        }
        switch (rank) {
```

```
            case 0:
                return 'A';
            case 9:
                return 'T';
            case 10:
                return 'J';
            case 11:
                return 'Q';
            case 12:
                return 'K';
        }
        return (char) rank;

    }

    private static void printDeck() {
        for (int i = 0; i < 52; i++) {
            int rank = i % 13;
            int suit = i / 13;
            System.out.println(getRank(rank) + " " + (char)
getSuit(suit));
        }
    }

    private static void initializeDeck() {
         for(int i = 0; i < 52; i++) {
            deck[i] = i;
        }
    }

    private static void shuffleDeck () {

        for(int i = 0; i < 10000; i++) {
            int card1 = randomInteger(0, 52);
            int card2 = randomInteger(0, 52);
            //swap card1 with card2
            int temp = deck[card1];
            deck[card1] = deck[card2];
            deck[card2] = temp;

        }
    }

    private static void dealPokerHand() {
        for (int i = 0; i < 5; i++) {
            //hand[i] = randomInteger(0, 52);
            hand[i] = deck[i];
        }

    }

    private static void displayPokerHand() {

        for (int i = 0; i < 5; i++) {
```

```
            char cardRank = getRank(hand[i] % 13);
            char cardSuit = (char) getSuit(hand[i] / 13);
            System.out.print(cardRank + "" + cardSuit);
            System.out.print(" ");
        }
        System.out.println("");

    }

    private static int hasPair() {
        for (int i = 0; i < 4; i++) {
            for (int j = i + 1; j > i && j < 5; j++) {
                if (hand[i] % 13 == hand[j] % 13) {
                    return hand[i];
                }
            }
        }
        return -1;
    }

    private static int hasThree() {
        //assumes the hand is sorted by rank
        int[] rank = new int[5];
        for (int i = 0; i < 5; i++) {
            rank[i] = hand[i] % 13;
        }
        for(int i = 0; i < 3; i++) {
            if(rank[i] == rank[i+1] && rank[i] == rank[i+2]){
                return hand[i];
            }
        }
        return -1;
    }

    private static void sortHand() {
        int[] rank = new int[5];
        for (int i = 0; i < 5; i++) {
            rank[i] = hand[i] % 13;
        }
        boolean flag = true;
        while (flag) {
            flag = false;
            for (int i = 0; i < 4; i++) {
                if (rank[i] > rank[i + 1]) {
                    int temp = rank[i];
                    rank[i] = rank[i + 1];
                    rank[i + 1] = temp;
                    int temp2 = hand[i];
                    hand[i] = hand[i + 1];
                    hand[i + 1] = temp2;
                    flag = true;
                }
            }
```

```
        }
    }
}
```

Java Project: QuadraticEquations

Description: This program solves randomly generated quadratic equations → $a x^2 + b x + c = 0$. Where a, b and c are integers between 0 and 10. This range can be easily changed in the program to solve other equations.

Sample Output:

run:
a = 5
b = 9
c = 3
The quadratic equation:
5x^2 + 9x + 3 = 0
Has solutions:
Real solutions:
x1 = -0.441742430504416 03
x2 = -1.3582575694955838
BUILD SUCCESSFUL (total time: 0 seconds)

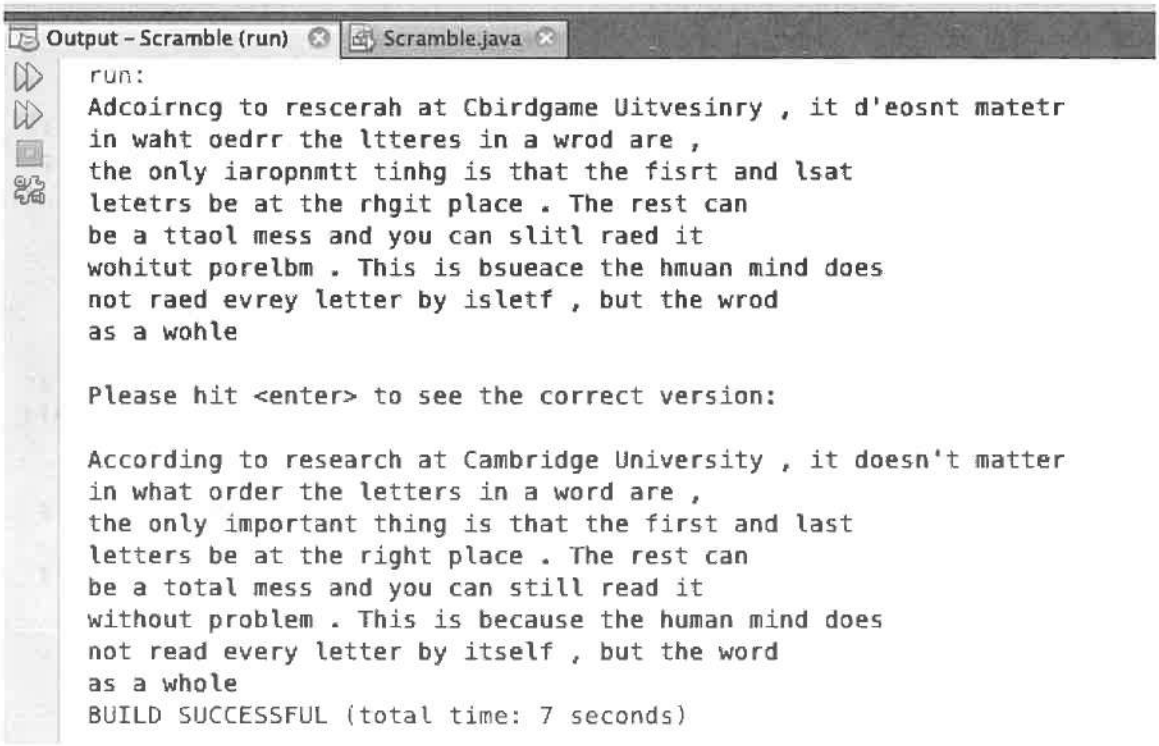
```
package quadraticequations;

public class QuadraticEquations {

    public static void main(String[] args) {
        //quadratic equation
        // x = (- b +- sqrt( b squared - 4 a c) )/2a
        //note: can't take sqrt of a negative number


            int a = randomInteger(0, 10);
            int b = randomInteger(0, 10);
            int c = randomInteger(0, 10);

            System.out.println("a = " + a);
            System.out.println("b = " + b);
            System.out.println("c = " + c);
```

```
        System.out.println("The quadratic equation:");

    System.out.println(a + "x^2 + " + b + "x + " + c + " = 0 ");

        System.out.println("Has solutions:");



        if (a == 0) {
            System.out.println("x = " + -c / b);
            System.exit(1);
        }
        double discriminant = Math.pow(b, 2) - 4 * a * c;
        if (discriminant >= 0) {
            double x1 = (-b + Math.sqrt(discriminant))/(2 * a);
            double x2 = (-b - Math.sqrt(discriminant))/(2 * a);
            System.out.println("Real solutions:");
            System.out.println("x1 = " + x1);
            System.out.println("x2 = " + x2);
        } else {
           double c1_real = -b / (2 * a);
           double c2_real = -b / (2 * a);
           double c1_imaginary=Math.sqrt(-discriminant)/(2* a);
           double c2_imaginary=-Math.sqrt(-discriminant)/(2* a);
           System.out.println("Complex solutions:");
           System.out.println("c1_real = " + c1_real);
           System.out.println("c1_imaginary = "+ c1_imaginary);
           System.out.println("c2_real = " + c2_real);
           System.out.println("c2_imaginary = "+ c2_imaginary);
        }



    }

    private static int randomInteger(int a, int b) {
        return (int) ((b - a) * Math.random() + a);
    }

}
```

Java Project: Scramble

Description: Scramble the words in a paragraph by shuffling the internal letters of each word, keeping the first and last letters in place.

Sample Output:



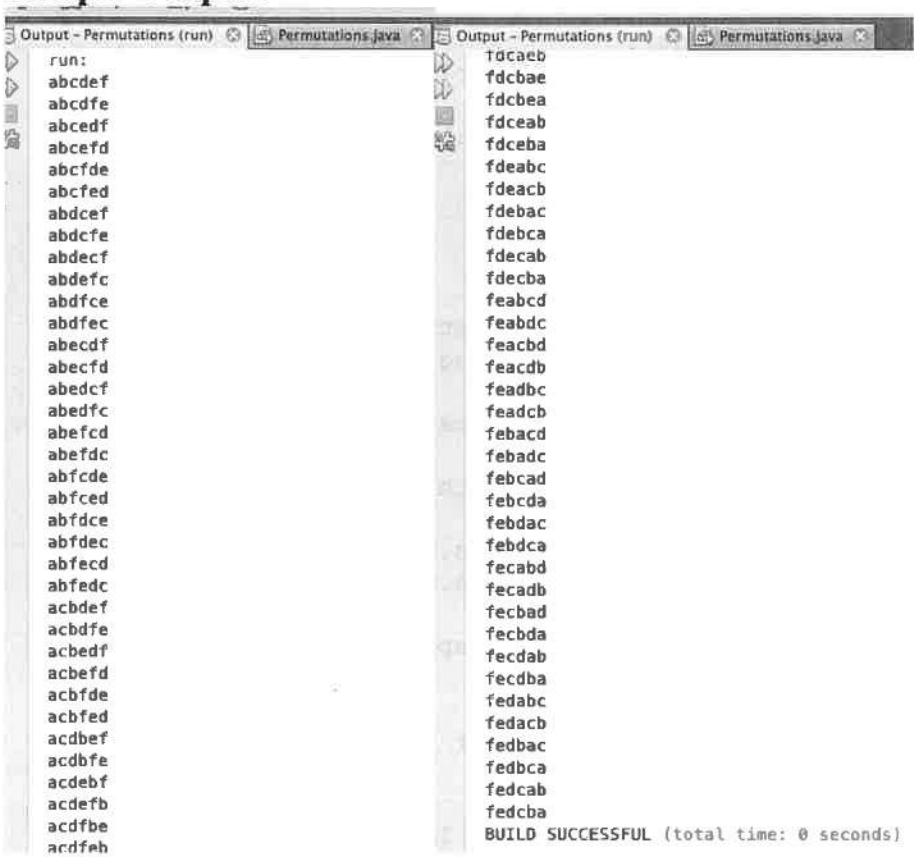

```
package scramble;

import java.util.Random;
import java.util.Scanner;

public class Scramble {

    public static void main(String[] args) {
        String sentence = "According to research at Cambridge
University , it doesn't matter \nin what order the letters in a word
are , \nthe only important thing is that the first and last
\nletters be at the right place . The rest can\nbe a total mess and
you can still read it \nwithout problem . This is because the human
mind does \nnot read every letter by itself , but the word\nas a
whole";

        String out = "";
        Scanner scanner = new Scanner(sentence);
```

```
        int count = 0;
        while (scanner.hasNext()) {
            String word = scanner.next();
            count++;
            out += innerScramble(word) + " ";
            if ((count % 10 ) == 0) out += "\n";
        }
        System.out.println(out);
        Scanner consoleInput = new Scanner(System.in);
        System.out.println("");
        System.out.println("Please hit <enter> to see the correct
version:");
        consoleInput.nextLine();
        System.out.println(sentence);

    }
//getScramble scrambles the entire input string
    public static String getScramble(String input) {
        String output = input;
        char[] outChars = output.toCharArray();
        Random r = new Random();
        for (int i = 0; i < outChars.length * outChars.length; i++)
{
            int swap1 = r.nextInt(outChars.length);
            int swap2 = r.nextInt(outChars.length);
            char temp = outChars[swap1];
            outChars[swap1] = outChars[swap2];
            outChars[swap2] = temp;
        }
        output = String.valueOf(outChars);
        return output;
    }
//innerScramble leaves the first and last letters in place.
    public static String innerScramble(String input) {
        String output = input;
        char[] outChars = output.toCharArray();
        if (outChars.length > 3) {
            Random r = new Random();
            for (int i = 0; i < outChars.length * outChars.length;
i++) {
                int swap1 = r.nextInt(outChars.length - 2) + 1;
                int swap2 = r.nextInt(outChars.length - 2) + 1;
                char temp = outChars[swap1];
                outChars[swap1] = outChars[swap2];
                outChars[swap2] = temp;
            }
        }
        output = String.valueOf(outChars);
        return output;
    }
}
```

Java Project: Permutations

Description: Print out all the permutations of a given string of letters.

Sample Output:




```
package permutations;
public class Permutations {
    public static void main(String[] args) {
        permutation("", "abcdef");
    }
    private static void permutation(String prefix, String str) {
        int n = str.length();
        if (n == 0) {
            System.out.println(prefix);
        } else {
            for (int i = 0; i < n; i++) {
                permutation(prefix + str.charAt(i), str.substring(0,
i) + str.substring(i + 1, n));
            }
        }
    }
}
```

Java Project: NumberGuessingGame

Description: Guess a hidden number between 1 and 1000. The program uses a do while loop to keep the iteration going until the user gets the correct number. The program gives the user a "higher/lower" hint after each guess.

Sample Output:

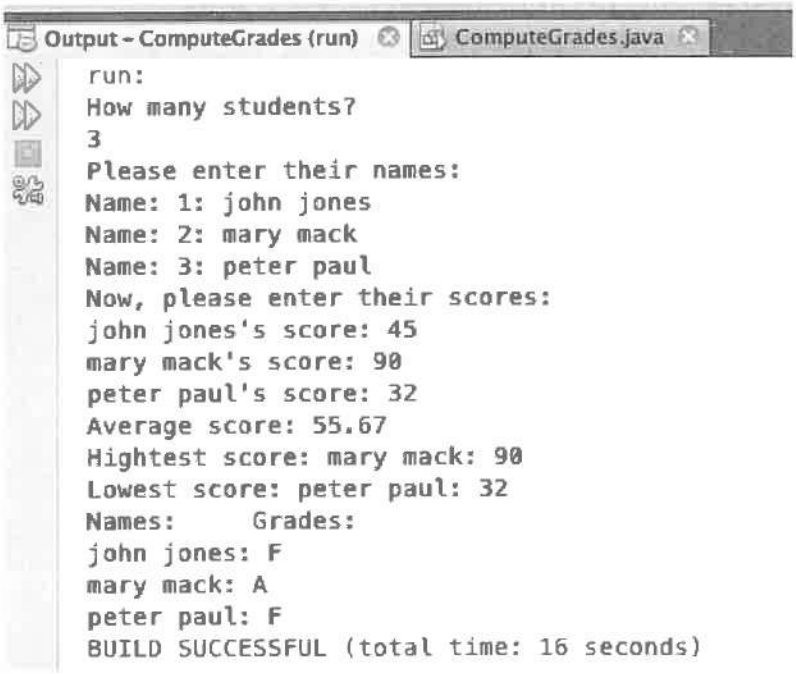
```
run:
Enter a guess (1-1000): 500
Your guess is greater than the secret number.
Enter a guess (1-1000): 250
Your guess is smaller than the secret number.
Enter a guess (1-1000): 325
Your guess is smaller than the secret number.
Enter a guess (1-1000): 400
Your guess is smaller than the secret number.
Enter a guess (1-1000): 450
Your guess is smaller than the secret number.
Enter a guess (1-1000): 475
Your guess is greater than the secret number.
Enter a guess (1-1000): 457
Your guess is greater than the secret number.
Enter a guess (1-1000): 453
Your guess is smaller than the secret number.
Enter a guess (1-1000): 455
Your guess is smaller than the secret number.
Enter a guess (1-1000): 456
Your guess is correct. Congratulations!
BUILD SUCCESSFUL (total time: 1 minute 1 second)
```

```
package numberguessinggame;

import java.util.Scanner;

public class NumberGuessingGame {

	public static void main(String[] args) {

		int secretNumber;
		secretNumber = (int) (Math.random() * 999 + 1);
		Scanner keyboard = new Scanner(System.in);
		int guess;
		do {
			System.out.print("Enter a guess (1-1000): ");
			guess = keyboard.nextInt();
			if (guess == secretNumber)
				System.out.println("Your guess is correct.
Congratulations!");
			else if (guess < secretNumber)
				System.out.println("Your guess is smaller
than the secret number.");
			else if (guess > secretNumber)
				System.out.println("Your guess is greater
than the secret number.");
		} while (guess != secretNumber);

	}

}
```

Java Project: ComputeGrades

Description: Help a teacher grade the students. The user enters the students' names and scores and the program computes their letter grades and other statistics.

Sample Output:




```
package computegrades;

import java.util.Scanner;

public class ComputeGrades {

    public static void main(String[] args) {
        String[] names = getNames();
        int[] scores = getscores(names);
        double average = computeAverage(scores);
        int highestIndex = getHighest(scores);
        int lowestIndex = getLowest(scores);
        System.out.format("Average score: %3.2f\n", average);
        System.out.println("Hightest score: " + names[highestIndex]
+ ": " + scores[highestIndex]);
        System.out.println("Lowest score: " + names[lowestIndex] +
": " + scores[lowestIndex]);
        System.out.println("Names:     Grades:    ");
```

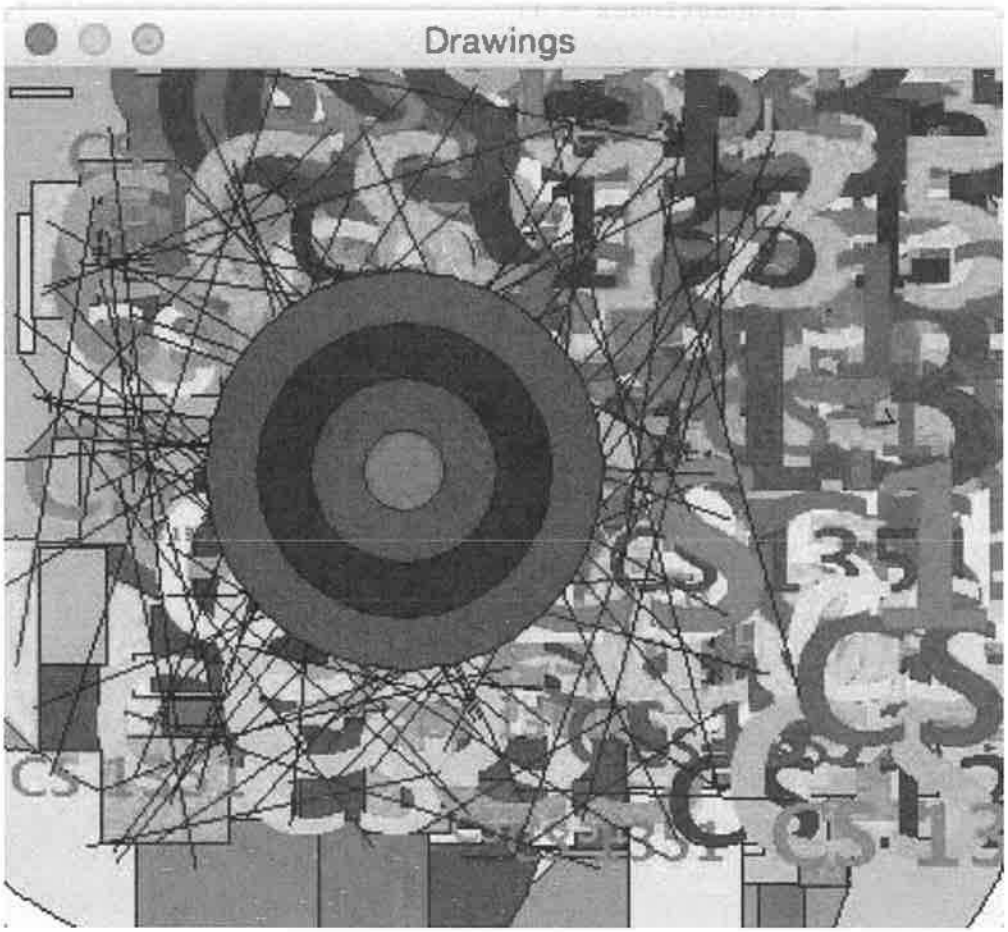
```
        System.out.print("");
        printLetterGrades(names, scores);
    }

    public static void printLetterGrades(String[] names, int[]
scores) {
        for (int i = 0; i < names.length; i++) {
            if (scores[i] >= 90) {
                System.out.println(names[i] + ": A");
            } else if (scores[i] >= 80) {
                System.out.println(names[i] + ": B");
            } else if (scores[i] >= 70) {
                System.out.println(names[i] + ": C");
            } else if (scores[i] >= 60) {
                System.out.println(names[i] + ": D");
            } else {
                System.out.println(names[i] + ": F");
            }
        }
    }

    public static String[] getNames() {
        Scanner input = new Scanner(System.in);
        System.out.println("How many students?");
        int n = input.nextInt();
        System.out.println("Please enter their names:");
        String[] names = new String[n];
        for (int i = 0; i < names.length; i++) {
            System.out.print("Name: " + (i + 1) + ": ");
            names[i] = input.next() + " " + input.next();
        }

        return names;
    }

    public static int[] getscores(String[] names) {
        System.out.println("Now, please enter their scores:");
        Scanner input = new Scanner(System.in);
        int[] scores = new int[names.length];
        for (int i = 0; i < names.length; i++) {
            System.out.print(names[i] + "'s ");
            System.out.print("score: ");
            while (!input.hasNextInt()) {
                input.next();
                System.out.println("Please enter an integer for the
score.");

            }
            scores[i] = input.nextInt();
        }
        return scores;
    }

    public static double computeAverage(int[] scores) {
```

```
        double sum = 0;
        for (int i = 0; i < scores.length; i++) {
            sum += scores[i];
        }
        return sum / scores.length;
    }

    public static int getHighest(int[] scores) {
        int highestIndex = Integer.MIN_VALUE;
        int highestScore = Integer.MIN_VALUE;
        for (int i = 0; i < scores.length; i++) {
            if (scores[i] > highestScore)
            {
                highestScore = scores[i];
                highestIndex = i;
            }
          }

        return highestIndex;
    }

    public static int getLowest(int[] scores) {
        int lowestIndex = Integer.MAX_VALUE;
        int lowestScore = Integer.MAX_VALUE;
        for (int i = 0; i < scores.length; i++) {
            if (scores[i] < lowestScore) {
                lowestScore = scores[i];
                lowestIndex = i;
            }
        }
        return lowestIndex;
    }
}
```

Java Project: RandomDrawings

Description: Draw a variety of objects to a drawing panel. This program is a demonstration of how to use Jave to draw Characers, Lines, Rectangles and circles.

Sample Output:



```
package randomdrawings;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class RandomDrawings extends JFrame {

    public static int W = 300;
    public static int H = 300;

    public RandomDrawings() {
        DrawingPanel drawingPanel = new DrawingPanel();
        this.add(drawingPanel);

    }

    static class DrawingPanel extends JPanel {

        public static double radius;
        public static double x;
        public static double y;
        public static Color color;

        public DrawingPanel() {
            radius = W / 8;
            x = W / 2;
            y = H / 2;
            color = new Color(100, 250, 10);

        }

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            drawCircles(g);
            drawRectangels(g);
            drawLetters(g);
            drawLines(g);
            drawConcentricCircles(g);
            //drawGrid(g);
        }

        public void drawGrid(Graphics g) {
            int cellW = 100;
            int cellH = 100;
            for(int i = 0; i < H/cellH; i++){
                for(int j = 0; j < W/cellW; j++) {
                    color = getRandomColor();
                    g.setColor(color);
                    g.fillRect(j*cellW, i*cellH, cellW, cellH);
                    g.setColor(Color.black);
```

```
                    g.drawRect(j*cellW, i*cellH, cellW, cellH);
                }
            }
        }

        public void drawConcentricCircles(Graphics g) {
            int minRadius = 0;
            int maxRadius = W/4;
            int centerX = W/2;
            int centerY = H/2;
            int delta = 20;
            for(int radius = maxRadius; radius > minRadius ; radius
-= delta) {
                color = getRandomColor();
                g.setColor(color);
                g.fillOval(
                        centerX - radius,
                        centerY - radius,
                        2*radius,
                        2*radius
                        );
                g.setColor(Color.black);
                g.drawOval(
                        centerX - radius,
                        centerY - radius,
                        2*radius,
                        2*radius
                        );
            }
        }

        public Color getRandomColor() {
            return color = new Color(
                    (int) (256 * Math.random()),
                    (int) (256 * Math.random()),
                    (int) (256 * Math.random())
            );

        }

        public void drawLines(Graphics g) {
            int n = 100;
            int x1, y1, x2, y2;
            for (int i = 0; i < n; i++) {
                color = getRandomColor();
                //g.setColor(color);
                x1 = (int) (Math.random() * W);
                y1 = (int) (Math.random() * H);
                x2 = (int) (Math.random() * W);
                y2 = (int) (Math.random() * H);
                g.drawLine(x1, y1, x2, y2);
                g.setColor(Color.black);

            }
```

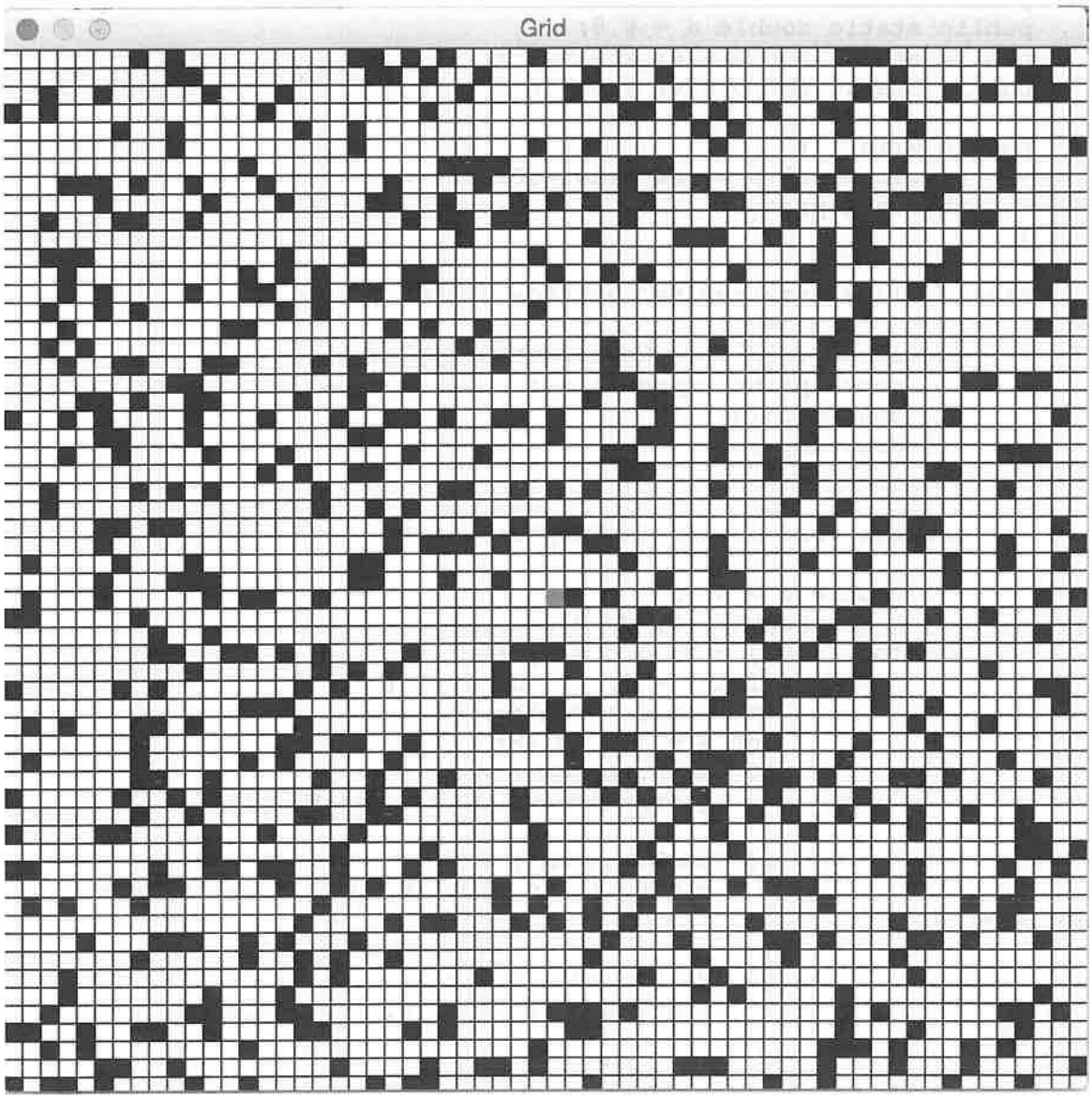
```
        }

        public void drawLetters(Graphics g) {

            int maxFontSize = 100;
            int n = 100;
            for (int i = 0; i < n; i++) {
                int fontSize = (int) (maxFontSize * Math.random());
                Font font = new Font("New Courier", Font.BOLD,
fontSize);

                g.setFont(font);
                String text = "CS 1351";
                getRandomColor();

                g.setColor(color);
                x = (int) (W * Math.random());
                y = (int) (H * Math.random());
                g.drawString(text, (int) x, (int) y);

            }

        }

        public void drawRectangels(Graphics g) {
            int maxRectWidth = W / 4;
            int maxRectHeight = H / 4;
            int n = 100;
            for (int i = 0; i < n; i++) {
                int rectWidth = (int) (maxRectWidth *
Math.random());
                int rectHeight = (int) (maxRectHeight *
Math.random());
                getRandomColor();
                x = (int) (W * Math.random());
                y = (int) (H * Math.random());
                g.setColor(color);
                g.fillRect((int) x, (int) y, rectWidth, rectHeight);
                g.setColor(Color.black);
                g.drawRect((int) x, (int) y, rectWidth, rectHeight);
            }
        }

        public void drawCircles(Graphics g) {
            int maxRadius = W / 4;

            int n = 50;
            for (int i = 0; i < n; i++) {
                radius = W / 2 * Math.random();
                getRandomColor();
                x = (int) (W * Math.random());
                y = (int) (H * Math.random());
                g.setColor(color);
                g.fillOval(
```

```
                        (int) (x - radius),
                        (int) (y - radius),
                        (int) (2 * radius),
                        (int) (2 * radius)
                );
                g.setColor(Color.black);
                g.drawOval(
                        (int) (x - radius),
                        (int) (y - radius),
                        (int) (2 * radius),
                        (int) (2 * radius)
                );

            }
        }

    }

    public static void main(String[] args) {
        RandomDrawings frame = new RandomDrawings();
        frame.setTitle("Drawings");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W, H);
        frame.setVisible(true);
    }

}
```

Java Project: Grid

Description: Draw a grid of boxes/cells that are filled, or not filled, depending on a random variable. Make one red cell to be the "player". There is no game play in this project but it is easy to imagine the red cell as a player moving around the grid (maybe with enemies moving around as well).

Sample Output:



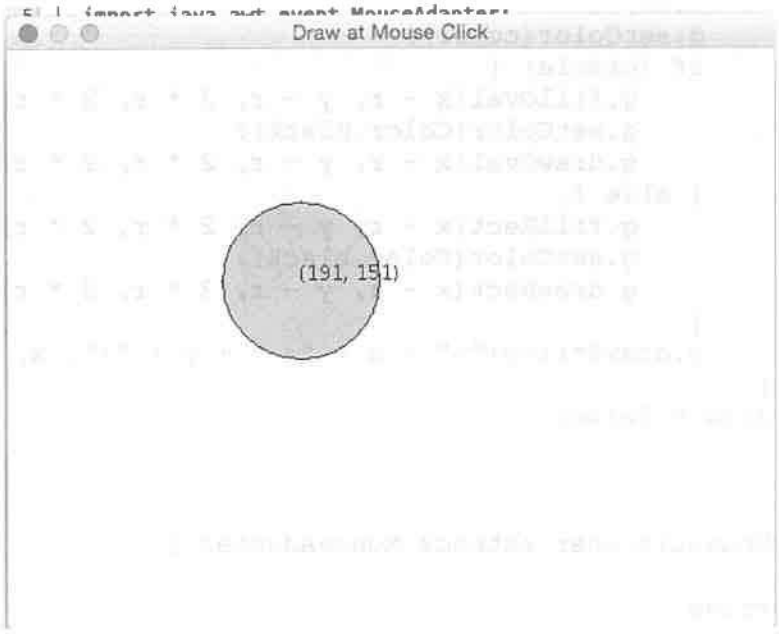
```
package grid;

import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;


public class Grid extends JFrame {

    public static int W = 600;
    public static int H = 600;
    public static int cellW = 10;
    public static int cellH = 10;
    public static int n = W / cellW;
    public static int m = H / cellH;
    public static double d = 0.8;
    public static int player_x = W / 2;
    public static int player_y = H / 2;

    public Grid() {
        DrawingPanel p = new DrawingPanel();
        this.add(p);
    }

    static class DrawingPanel extends JPanel {

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            drawGrid3(g);
            drawPlayer(g);
        }

        private void drawGrid1(Graphics g) {

            double d = 0.8;
            g.drawRect(0, 0, W, H);
            for (int j = 0; j < m; j++) {
                for (int i = 0; i < n; i++) {
                    int x = i * cellW;
                    int y = j * cellH;
                    if (Math.random() > d) {
                        g.drawLine(x, y, x + cellW, y);
                    }
                    if (Math.random() > d) {
                        g.drawLine(x + cellW, y, x + cellW, y +
cellH);
                    }
                    if (Math.random() > d) {
                        g.drawLine(x + cellW, y + cellH, x, y +
cellW);
                    }
                    if (Math.random() > d) {
                        g.drawLine(x, y + cellW, x, y);
```

```
                }
            }
        }
    }

    private void drawGrid2(Graphics g) {

        g.drawRect(0, 0, W, H);
        for (int j = 0; j < m; j++) {
            for (int i = 0; i < n; i++) {
                int x = i * cellW;
                int y = j * cellH;
                //g.drawRect(x, y, cellW, cellH);
                if (Math.random() > d) {
                    g.drawLine(x, y, x + cellW, y);
                }
                if (Math.random() > d) {
                    g.drawLine(x, y + cellW, x, y);
                }
            }
        }
    }

    private void drawGrid3(Graphics g) {
        g.setColor(Color.black);
        g.drawRect(0, 0, W, H);
        for (int j = 0; j < m; j++) {
            for (int i = 0; i < n; i++) {
                int x = i * cellW;
                int y = j * cellH;
                g.drawRect(x, y, cellW, cellH);
                if ( Math.random() > d) {
                    g.fillRect(x, y, cellW, cellH);
                }
            }
        }
    }

    public static void drawPlayer(Graphics g) {
        g.setColor(Color.red);
        g.fillRect(player_x, player_y, cellW, cellH);
    }
}

public static void main(String[] args) {
    Grid frame = new Grid();
    frame.setTitle("Grid");
    frame.setSize(W, H);
    frame.setLocationRelativeTo(null);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
}

}
```

Java Project: MouseClickDraw

Description: Interactive circle/square drawing program. When the user clicks (left) in the drawing window a circle is draw of random radius and color. If the user right-clicks the circle changes to another random color. A middle-click turns the circle into a square.

Sample Output:




```
package mouseclickdraw;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Random;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class MouseClickDraw extends JFrame {

    public boolean draw = false;
    public int x;
    public int y;
    public int r;
    public Color color;
```

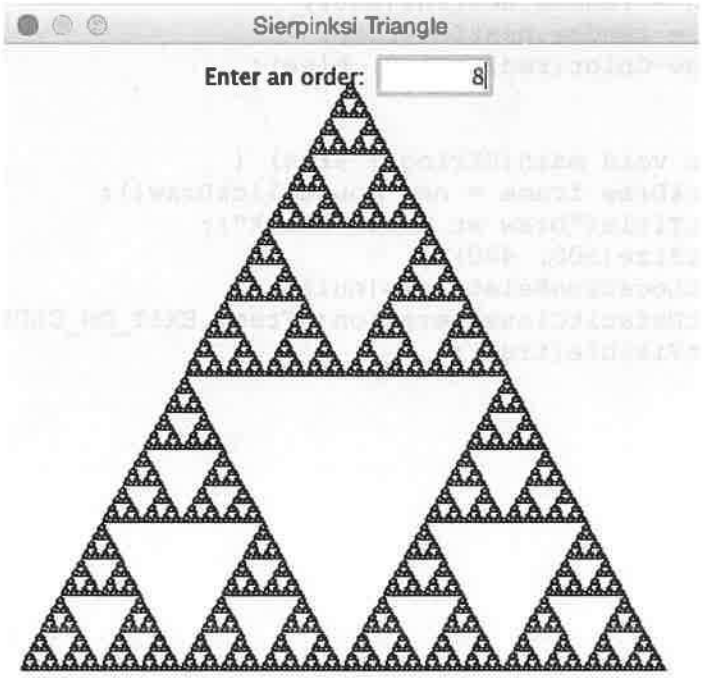
```
public boolean circle = true;

public MouseClickDraw() {
    JPanel panel = new DrawingPanel();
    panel.addMouseListener(new myMouseListener());
    panel.setFocusable(true);
    add(panel);
}

public class DrawingPanel extends JPanel {

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        if (draw) {
            g.setColor(color);
            if (circle) {
                g.fillOval(x - r, y - r, 2 * r, 2 * r);
                g.setColor(Color.black);
                g.drawOval(x - r, y - r, 2 * r, 2 * r);
            } else {
                g.fillRect(x - r, y - r, 2 * r, 2 * r);
                g.setColor(Color.black);
                g.drawRect(x - r, y - r, 2 * r, 2 * r);
            }
            g.drawString("(" + x + ", " + y + ")", x, y);
        }
        draw = false;
    }
}

class myMouseListener extends MouseAdapter {

    @Override
    public void mouseClicked(MouseEvent e) {

        if (e.getButton() == e.BUTTON1) {
            draw = true;
            x = e.getX();
            y = e.getY();
            r = (int) (100 * Math.random());
            repaint();
        }

        if (e.getButton() == e.BUTTON2) {
            draw = true;
            if (circle) {
                circle = false;
            } else {
                circle = true;
            }
            repaint();
        }
```

```
            if (e.getButton() == e.BUTTON3) {
                draw = true;
                color = getRandomColor();
                repaint();
            }
        }
    }

    public Color getRandomColor() {
        Random random = new Random();
        int red = random.nextInt(256);
        int green = random.nextInt(256);
        int blue = random.nextInt(256);
        return new Color(red, green, blue);
    }

    public static void main(String[] args) {
        MouseClickDraw frame = new MouseClickDraw();
        frame.setTitle("Draw at Mouse Click");
        frame.setSize(500, 400);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Java Project: Sierpinski

Description: Draw a triangle-fractal by sequentially drawing triangles based on the halfway point on the sides of a give triangle.

Sample Output:



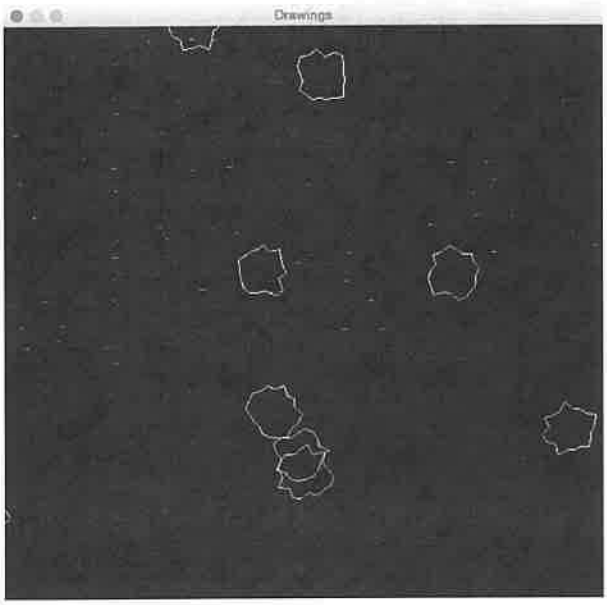
```
package sierpinski;

import java.awt.BorderLayout;
import java.awt.Graphics;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class Sierpinski extends JFrame {

    public Sierpinski() {
        JPanel panel = new SierpinskiPanel();
        add(panel, new BorderLayout().CENTER);
    }

    static class SierpinskiPanel extends JPanel {

        public static int order = 0;
        private JTextField orderField = new JTextField("0", 5);

        SierpinskiPanel() {
            add(new JLabel("Enter an order:"));
            add(orderField);
            orderField.setHorizontalAlignment(SwingConstants.RIGHT);

            orderField.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {

setOrder(Integer.parseInt(orderField.getText()));
                }
            });
        }

        public void setOrder(int order) {
            this.order = order;
            repaint();
        }

        protected void paintComponent(Graphics g) {
            super.paintComponents(g);
            //the "20" is a padding so the drawing deosnt go to the
edge of the window
            Point p1 = new Point(getWidth() / 2, 20);
            Point p2 = new Point(10, getHeight() - 20);
            Point p3 = new Point(getWidth() - 20, getHeight() - 20);
            displayTriangles(g, order, p1, p2, p3);
        }
```

```
        private static void displayTriangles(Graphics g, int order,
Point p1, Point p2, Point p3) {
            if (order == 0) {
                g.drawLine(p1.x, p1.y, p2.x, p2.y);
                g.drawLine(p2.x, p2.y, p3.x, p3.y);
                g.drawLine(p3.x, p3.y, p1.x, p1.y);
            } else {
                Point p12 = midpoint(p1, p2);
                Point p13 = midpoint(p1, p3);
                Point p23 = midpoint(p2, p3);
                displayTriangles(g, order - 1, p1, p12, p13);
                displayTriangles(g, order - 1, p2, p12, p23);
                displayTriangles(g, order - 1, p3, p13, p23);
            }
        }

        public static Point midpoint(Point p1, Point p2) {
            return new Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2);
        }
    }

    public static void main(String[] args) {
        Sierpinski sierpinski = new Sierpinski();
        sierpinski.setSize(400, 400);
        sierpinski.setTitle("Sierpinksi Triangle");
        sierpinski.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        sierpinski.setLocationRelativeTo(null);
        sierpinski.setVisible(true);
    }

}
```

Java Project: Asteroids

Description: Creates black/white asteroid environment. Great start for a shooting game similar to the original asteroids game.

Sample Output:




```
package asteroids;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

public class Asteroids extends JFrame {

    public static int W = 600;
    public static int H = 600;

    public Asteroids() {
        DrawingPanel drawingPanel = new DrawingPanel();
        drawingPanel.setBackground(Color.black);
        this.add(drawingPanel);
    }

    static class DrawingPanel extends JPanel {
```

```
        private Timer timer = new Timer(30, new TimerListener());
        int m = 20; //points per poly
        int n = 10; // number of polys
        double[][] x = new double[n][m];
        double[][] y = new double[n][m];
        double[] dx = new double[n];
        double[] dy = new double[n];
        double[] speed = new double[n];
        double maxRadius = 30;
        double minRadius = 20;
        double[] xCenter = new double[n];
        double[] yCenter = new double[n];
        public Color[] color = new Color[n];
        public double maxSpeed = 5;

        public DrawingPanel() {
            initializePolygons();
            timer.start();
        }

        private void initializePolygons() {
            double theta;
            for (int i = 0; i < n; i++) {
                color[i] = randomColor();
                xCenter[i] = Math.random() * W;
                yCenter[i] = Math.random() * H;
                for (int k = 0; k < m; k++) {
                    theta = k * 2.0 * Math.PI / m;
                    double r = minRadius + (maxRadius - minRadius) *
Math.random();
                    //use this line if you want regular polygons:
                    //double r = maxRadius;
                    x[i][k] = r * Math.cos(theta);
                    y[i][k] = r * Math.sin(theta);

                }
                speed[i] = maxSpeed;
                dx[i] = 2 * speed[i] * Math.random() - speed[i];
                dy[i] = 2 * speed[i] * Math.random() - speed[i];
            }
        }

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            updatePositions();
            checkBoundary();
            drawPolygons(g);
        }

        public void checkBoundary() {
            for (int i = 0; i < n; i++) {
                if (xCenter[i] + maxRadius < 0) {
                    xCenter[i] = getWidth() + maxRadius;
```

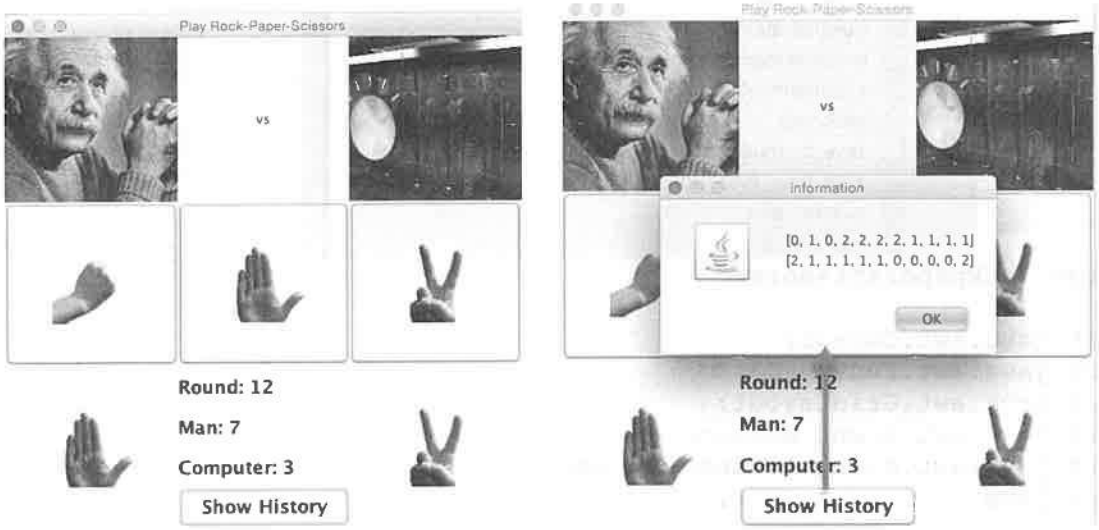
```
                    //dx[i] = Math.abs(dx[i]);
                }
                if (xCenter[i] - maxRadius > getWidth()) {
                    xCenter[i] = -maxRadius;
                    //dx[i] = -Math.abs(dx[i]);
                }
                if (yCenter[i] + maxRadius < 0) {
                    yCenter[i] = getHeight() + maxRadius;
                    //dy[i] = Math.abs(dy[i]);
                }
                if (yCenter[i] - maxRadius > getHeight()) {
                    yCenter[i] = -maxRadius;
                    //dy[i] = -Math.abs(dy[i]);
                }

            }
        }

        private void updatePositions() {
            double alpha = 0.1;
            for (int i = 0; i < n; i++) {
                for (int k = 0; k < m; k++) {

                    double newX = (x[i][k]) * Math.cos(alpha) +
(y[i][k]) * Math.sin(alpha);
                    double newY = -(x[i][k]) * Math.sin(alpha) +
(y[i][k]) * Math.cos(alpha);

                    x[i][k] = newX;
                    y[i][k] = newY;
                }
                xCenter[i] += dx[i];
                yCenter[i] += dy[i];
            }

        }

        class TimerListener implements ActionListener {

            public void actionPerformed(ActionEvent e) {
                repaint();
            }
        }

        private void drawPolygons(Graphics g) {
            for (int i = 0; i < n; i++) {
                int[] polyX = new int[m];
                int[] polyY = new int[m];
                //Color color = randomColor();
                g.setColor(color[i]);
                for (int k = 0; k < m; k++) {
                    polyX[k] = (int) (x[i][k] + xCenter[i]);
                    polyY[k] = (int) (y[i][k] + yCenter[i]);
                }
```

```
                //g.fillPolygon(polyX, polyY, m);
                g.setColor(Color.white);
                g.drawPolygon(polyX, polyY, m);
            }
        }

        private Color randomColor() {
            Color color = new Color(
                    (int) (256 * Math.random()),
                    (int) (256 * Math.random()),
                    (int) (256 * Math.random())
            );
            return color;
        }
    }

    public static void main(String[] args) {
        Asteroids frame = new Asteroids();
        frame.setTitle("Drawings");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W, H);
        frame.setVisible(true);
    }
}
```

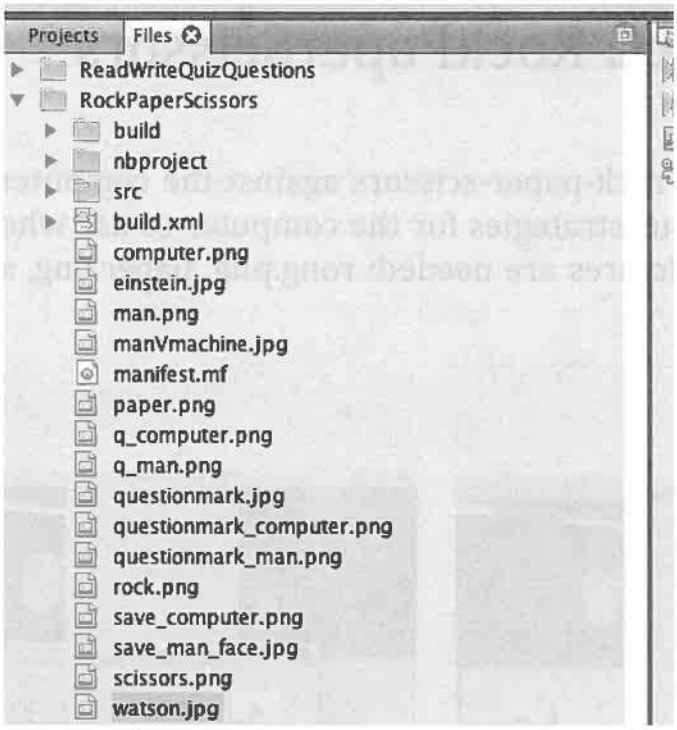
Java Project: RockPaperScissors

Description: Play rock-paper-scissors against the computer. The user can program various strategies for the computer to use when playing a human. Several pictures are needed: rong.png, paper.png, scissors.png etc.

Sample Output:



Need images:




```
package rockpaperscissors;

import java.awt.Cursor;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import javax.swing.JPanel;

public class RockPaperScissors extends JFrame {

    ImageIcon rockImage = new ImageIcon("rock.png");
    ImageIcon paperImage = new ImageIcon("paper.png");
    ImageIcon scissorsImage = new ImageIcon("scissors.png");
    ImageIcon manImage = new ImageIcon("einstein.jpg");
    ImageIcon computerImage = new ImageIcon("watson.jpg");

    JButton rock = new JButton(rockImage);
    JButton paper = new JButton(paperImage);
    JButton scissors = new JButton(scissorsImage);

    JLabel manLabel = new JLabel(manImage);
    JLabel computerLabel = new JLabel(computerImage);
```

```
JLabel middle1 = new JLabel("vs");

JLabel new_manMove = new JLabel("(man move)");
JLabel new_computerMove = new JLabel("(computer move)");

int playerMove = -1;
int computerMove = -1;
int[][] payoff = new int[3][3];
int result = 0;
int total = 0;
int n = 1;
int manCount = 0;
int computerCount = 0;
String historyString = "";
ArrayList resultsHistory;
ArrayList playerHistory;
ArrayList computerHistory;
JLabel round = new JLabel("Round: " + n);
JLabel manScore = new JLabel("Man: " + manCount);
JLabel computerScore = new JLabel("Computer: " + computerCount);
JButton history = new JButton("Show History");

public RockPaperScissors() {
    resultsHistory = new ArrayList<>();
    playerHistory = new ArrayList<>();
    computerHistory = new ArrayList<>();
    JPanel newPanel = new JPanel(new GridLayout(3, 3));
    newPanel.setCursor(new Cursor(Cursor.HAND_CURSOR));

    newPanel.add(manLabel);
    newPanel.add(middle1);
    newPanel.add(computerLabel);
    newPanel.add(rock);
    newPanel.add(paper);
    newPanel.add(scissors);

    RPSListener rpsListener = new RPSListener();
    rock.addActionListener(rpsListener);
    paper.addActionListener(rpsListener);
    scissors.addActionListener(rpsListener);

    initializePayoff(payoff);

    middle1.setHorizontalAlignment(JLabel.CENTER);
    new_manMove.setHorizontalAlignment(JLabel.CENTER);
    new_computerMove.setHorizontalAlignment(JLabel.CENTER);

    //This is the lower, middle, panel:
    JPanel middle2 = new JPanel(new GridLayout(4, 1));
    Font f = new Font("New Courier", Font.BOLD, 18);
    round.setFont(f);
    manScore.setFont(f);
    computerScore.setFont(f);
    history.setFont(f);
```

```
        middle2.add(round);
        middle2.add(manScore);
        middle2.add(computerScore);
        middle2.add(history);
        HistoryListener historyListener = new HistoryListener();
        history.addActionListener(historyListener);

        newPanel.add(new_manMove);
        newPanel.add(middle2);
        newPanel.add(new_computerMove);

        this.add(newPanel);
    }

    class HistoryListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(null, historyString,
"information", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    class RPSListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            new_manMove.setText("");
            new_computerMove.setText("");
            if (e.getSource() == rock) {
                playerMove = 0;
                new_manMove.setIcon(rockImage);
            }
            if (e.getSource() == paper) {
                playerMove = 1;
                new_manMove.setIcon(paperImage);
            }
            if (e.getSource() == scissors) {
                playerMove = 2;
                new_manMove.setIcon(scissorsImage);
            }
            //Get the computer's move:
            computerMove = getComputerMove2(playerHistory);
            if (computerMove == 0) {
                new_computerMove.setIcon(rockImage);
            }

            if (computerMove == 1) {
                new_computerMove.setIcon(paperImage);
            }

            if (computerMove == 2) {
                new_computerMove.setIcon(scissorsImage);
            }

            result = payoff[playerMove][computerMove];
```

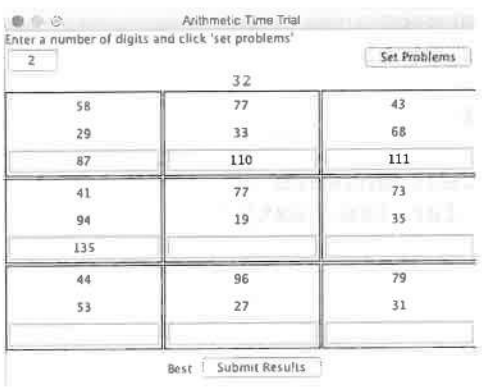
```
            if (result == 1) {
                //Man wins:
                manCount++;
            }

            if (result == -1) {
                //Computer wins:
                computerCount++;
            }
            playerHistory.add(playerMove);
            computerHistory.add(computerMove);
            historyString = playerHistory.toString();
            historyString += "\n" + computerHistory.toString();
            resultsHistory.add(result);
            total += result;
            n++;
            round.setText("Round: " + n);
            manScore.setText("Man: " + manCount);
            computerScore.setText("Computer: " + computerCount);

        }
    }

    private static void initializePayoff(int[][] payoff) {
        //player move: Rock
        payoff[0][0] = 0;   //computer move: Rock
        payoff[0][1] = -1;  //computer move: Paper
        payoff[0][2] = 1;   //computer move: Scissors

        //player move: Paper
        payoff[1][0] = 1;   //computer move: Rock
        payoff[1][1] = 0;   //computer move: Paper
        payoff[1][2] = -1;  //computer move: Scissors

        //Player move: Scissors
        payoff[2][0] = -1;  //computer move: Scissors
        payoff[2][1] = 1;   //computer move: Scissors
        payoff[2][2] = 0;   //computer move: Scissors
    }

    private static int getComputerMove1() {
        //The computer is making a random move:
        Random rand = new Random();
        return rand.nextInt(3);
    }

    private static int getComputerMove2(ArrayList<Integer>
playerHistory) {
        //The computer is trying to be intelligent.
        //It looks at the player history and calcuates the most
frequent
        //player move, and then assumes the player will make that
move
```

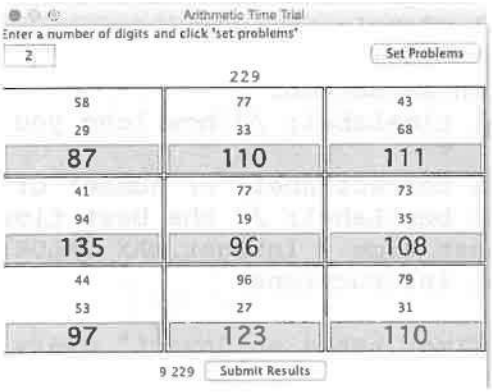
```
        //and plays against it.
        Random rand = new Random();
        int mostCommonMove;

        int[] counter = new int[3];
        for (int i = 0; i < playerHistory.size(); i++) {
            counter[playerHistory.get(i)]++;
        }

        if (counter[0] == counter[1] && counter[1] == counter[2]) {
            // if all counts are equal, random move:
            return rand.nextInt(3);
        } else {
            mostCommonMove = getMaxIndex(counter);
        }
        switch (mostCommonMove) {
            case 0: //if most common move is Rock
                return 1; //play Paper
            case 1: //if most common move is Paper
                return 2; //play Scissors
            case 2: //if most common move is Scissors
                return 0; //play Rock
        }
        return 0;
    }

    private static int getMaxIndex(int[] counter) {
        //This does not account for ties.
        //if there is a tie, it returns the first one found.
        int max = -1, max_index = -1;
        for (int i = 0; i < counter.length; i++) {
            if (counter[i] > max) {
                max = counter[i];
                max_index = i;
            }
        }
        return max_index;
    }

    public static void main(String[] args) {
        JFrame frame = new RockPaperScissors();
        frame.setTitle("Play Rock-Paper-Scissors");
        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

Java Project: ArithmeticGUI

Description: Solve arithmetic problems as fast as you can, with a GUI to help layout the problems in a graphical grid.

Sample Output:






```
//Melissa Wood
//CS 1351 D10
//Arithmetic GUI
package arithmeticgui;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.Timer;


public class ArithmeticGUI extends JFrame implements ActionListener
{

    int p = 3; // problems are p x q
    int q = 3;
```

```
    int m = 3; // number of digits, default value

    long start = System.currentTimeMillis();
    JButton submit;

    JLabel[][] add1 = new JLabel[p][q]; // the first number to add
    JLabel[][] add2 = new JLabel[p][q]; // the second number to add
    JTextField[][] result = new JTextField[p][q]; // your answer in
the text field
    int[][] correct_ans = new int[p][q]; // correct answer
    int[][] your_ans = new int[p][q]; // your answer

    JLabel mLabel;
    JTextField mTextField; // for the number of digits
    JButton setButton;
    JLabel timeLabel; // how long you took

    JLabel correctLabel; // number of correct answers
    JLabel bestLabel; // the best time so far (as text)
    int best_time = Integer.MAX_VALUE; // best time so far (int)
    JLabel instructions;

 // this timer sends an "event" every 1000 milliseconds (1 second)
to TimerListener
    Timer timer = new Timer(1000, new TimerListener());

    public ArithmeticGUI() {
        // BorderLayout for the JFrame:
        this.setLayout(new BorderLayout());
        // add three panels:
        createTopPanel(); // the "settings" panel
        createMiddlePanel(); // the "problems" panel
        createBottomPanel(); // the "results" panel
    }

    public void createBottomPanel() {
        // for the "results" panel (Bottom Panel):
        JPanel resultsPanel = new JPanel(new FlowLayout());
        resultsPanel.setBackground(Color.white);
        correctLabel = new JLabel("Number Correct");
        bestLabel = new JLabel("Best");
        submit = new JButton("Submit Results");
        submit.addActionListener(this);
        resultsPanel.add(correctLabel);
        resultsPanel.add(bestLabel);
        resultsPanel.add(submit);
        this.add(resultsPanel, BorderLayout.SOUTH);
     }

    public void createTopPanel() {
        // for the "settings" panel (Top Panel):
        JPanel settingsPanel = new JPanel(new BorderLayout());
        settingsPanel.setBackground(Color.white);
```

```
        mTextField = new JTextField();
        mTextField.setHorizontalAlignment(SwingConstants.CENTER);
        mTextField.setColumns(4);

        setButton = new JButton("Set Problems");
        setButton.addActionListener(this);

        timeLabel = new JLabel("", SwingConstants.CENTER);
        Font f = new Font("Courier", Font.BOLD, 18);
        timeLabel.setFont(f);
        timeLabel.setForeground(Color.red);
        timeLabel.setText("0");

        String instructions = "Enter a number of digits and click
'set problems'";
        JLabel instructionsLabel = new JLabel(instructions,
SwingConstants.LEFT);

        settingsPanel.add(instructionsLabel, BorderLayout.NORTH);
        settingsPanel.add(mTextField, BorderLayout.WEST);
        settingsPanel.add(setButton, BorderLayout.EAST);
        settingsPanel.add(timeLabel, BorderLayout.SOUTH);
        this.add(settingsPanel, BorderLayout.NORTH);
    }

    public void createMiddlePanel() {
        // This is the Middle panel
        JPanel problemsPanel = new JPanel(new GridLayout(q, p, 2,
2));

        for (int i = 0; i < p; i++) {
            for (int j = 0; j < q; j++) {
                JPanel problem = new JPanel(new GridLayout(3, 1, 0,
0));

problem.setBorder(BorderFactory.createLineBorder(Color.black));

                add1[i][j] = new JLabel("**",
SwingConstants.CENTER);
                problem.add(add1[i][j]);

                add2[i][j] = new JLabel("**",
SwingConstants.CENTER);
                problem.add(add2[i][j]);

                   result[i][j] = new JTextField(3);

result[i][j].setHorizontalAlignment(JTextField.CENTER);

                   problem.add(result[i][j]);
                   problemsPanel.add(problem);
                }
            }
            problemsPanel.setBackground(Color.yellow);
```

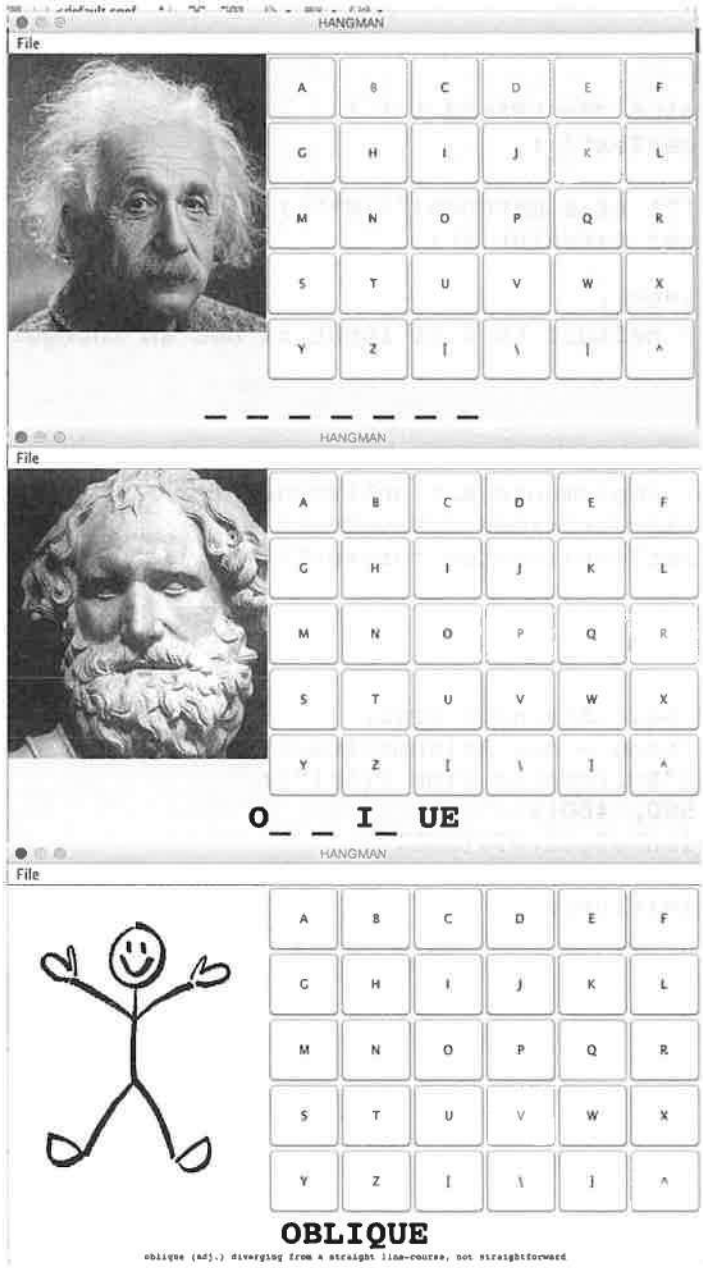
```
            this.add(problemsPanel, BorderLayout.CENTER);
        }

    public static int getRandomInt(int a, int b) {
        return (int) ((b + 1 - a) * Math.random() + a);
    }

    @Override

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == submit) {
            timer.stop();
            long stop = System.currentTimeMillis();
            timeLabel.setText("" + (stop - start) / 1000);
            int count = 0; // number right
            Color correctColor = new Color(200, 200, 255);
            Color wrongColor = new Color(255, 200, 200);
            for (int i = 0; i < p; i++) {
                for (int j = 0; j < q; j++) {
                    your_ans[i][j] = getIntValue(result[i][j]);
                    if (your_ans[i][j] == correct_ans[i][j]) {
                        result[i][j].setBackground(correctColor);
                        result[i][j].setFont(new Font("Geneva",
Font.BOLD, 24));
                        count++;
                    } else {
                      result[i][j].setBackground(wrongColor);
                    }
                }
            }
            correctLabel.setText("" + count);
            if ((Integer.parseInt(correctLabel.getText()) == p * q)
                  && (Integer.parseInt(timeLabel.getText()) <
best_time)) {
            best_time = Integer.parseInt(timeLabel.getText());
            bestLabel.setText("" + best_time);
            }
        }
        if (e.getSource() == setButton) {
            resetProblems();
        }
    }
    public void resetProblems() {
        m = getIntValue(mTextField);

        correctLabel.setText("");
        timeLabel.setText("");

        int a = (int) Math.pow(10, m - 1);
        int b = (int) Math.pow(10, m) - 1;

        for (int i = 0; i < p; i++) {
            for (int j = 0; j < q; j++) {
            int x1 = getRandomInt(a, b);
```

```
            add1[i][j].setText("" + x1);
            int x2 = getRandomInt(a, b);
            add2[i][j].setText("" + x2);
            correct_ans[i][j] = x1 + x2;
            result[i][j].setText("");
            result[i][j].setBackground(Color.white); // added to
reset fields to white for new set of problems.
            }
        }
        start = System.currentTimeMillis();
        timer.start();

    }
    public int getIntValue(JTextField in) {
        String s = in.getText();
        int out = 0;
        if (!s.equals("") && s.matches("\\d*")) {
            out = Integer.parseInt(s);
        } else {
            in.setText("0");
            out = 0; // default to 0 if input is not an integer
        }
    return out;
    }

    class TimerListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            timeLabel.setText((System.currentTimeMillis() - start) /
1000 + "");
        }
    }

    public static void main(String[] args) {
        ArithmeticGUI frame = new ArithmeticGUI();
        frame.setTitle("Arithmetic Time Trial");
        frame.setSize(500, 400);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Java Project: HangmanNew

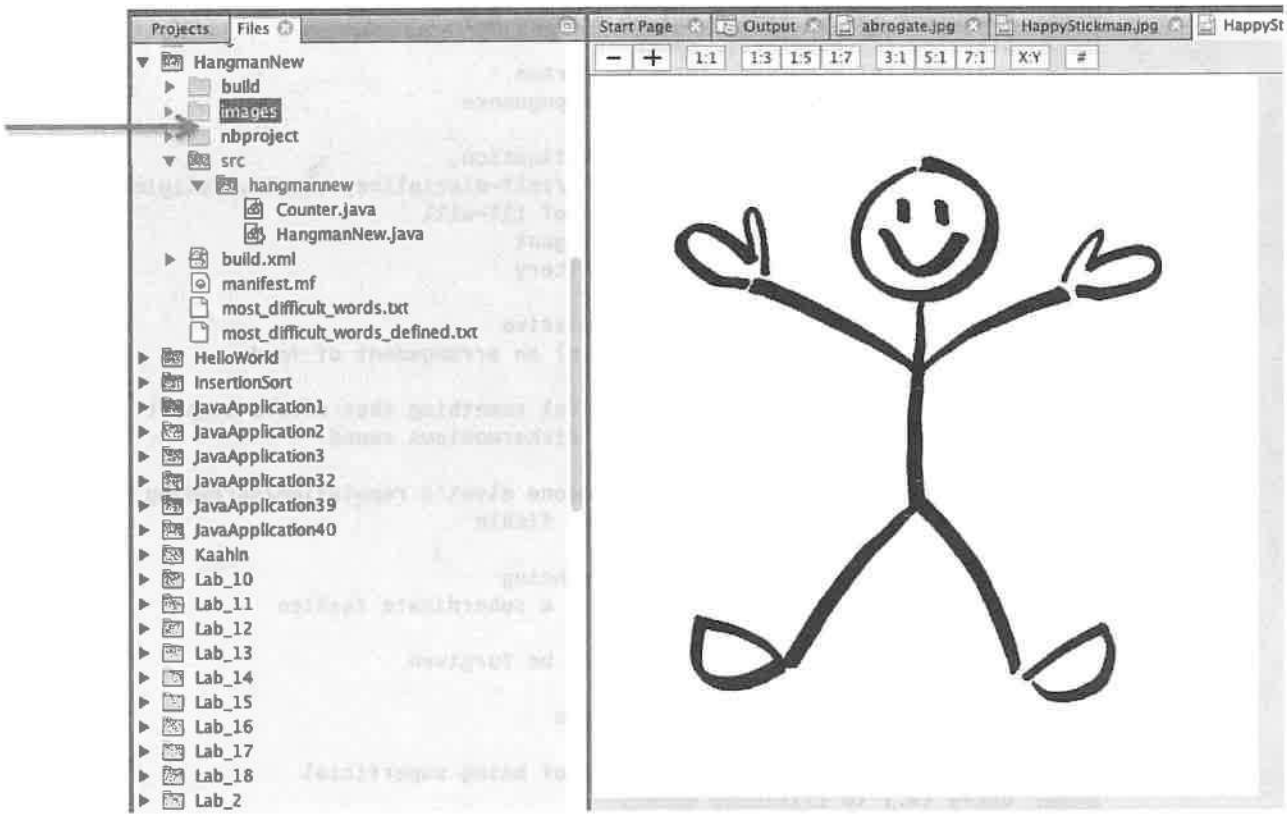
Description: Play hangman using a challenging set of words.

Sample Output:



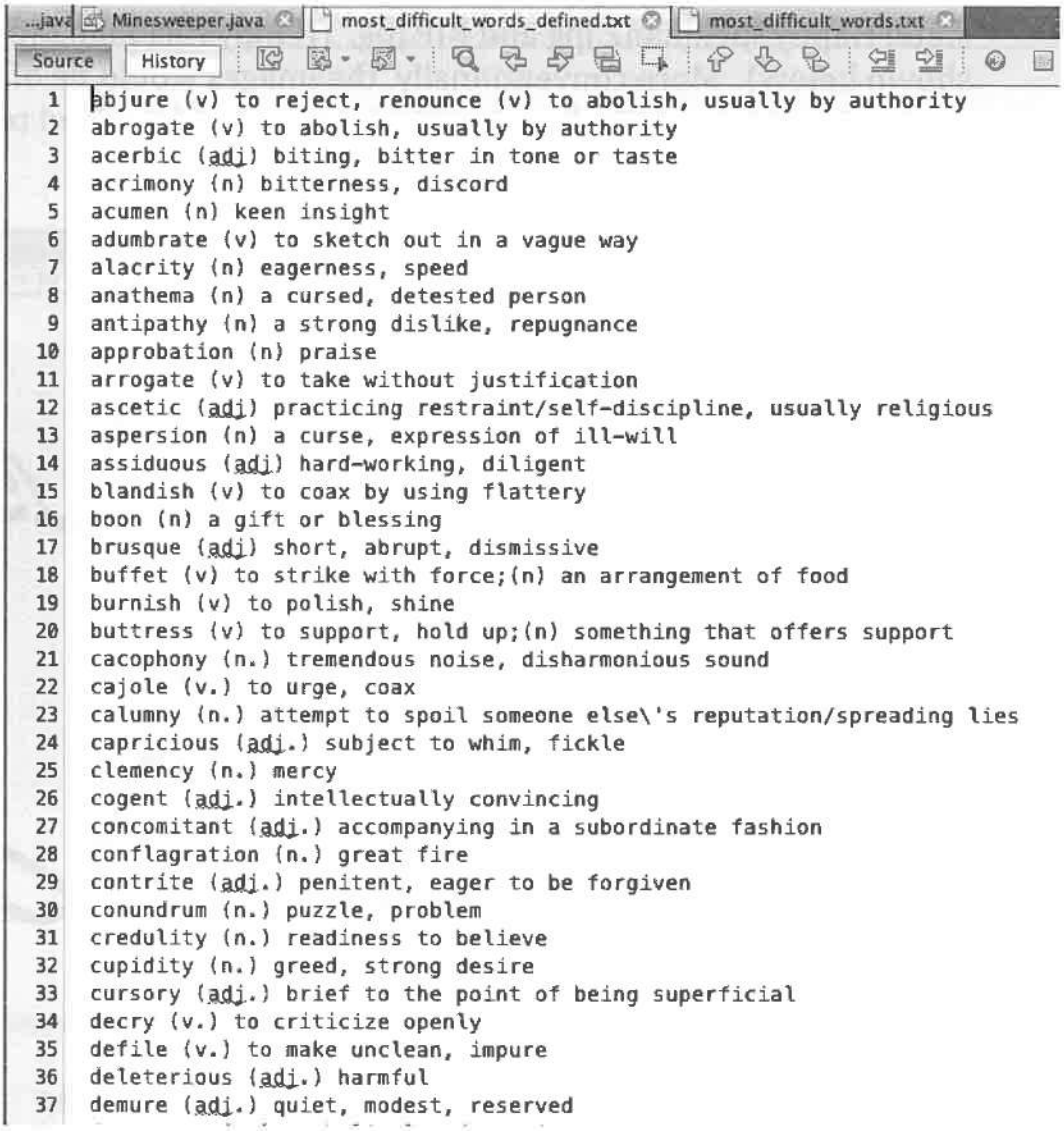
Need to include images in an images directory, named image0.jpg, image1.jpg, ..., image12.jpg. Also, images for the "win" and "lose"

state: HappyStickman2.jpg and RIP.jpg. (HappyStickman2.jpg is shown below). More conventionally, the images would be of successive stages of the hanging of stickman, but I decided to substitute pictures of famous scientists, geek that I am .



Notice also, the Counter.java class that is included with the HangmanNew.java main class.

Also, need the files: most_difficult_words_defined.txt, as shown in the next image:




```
1  abjure (v) to reject, renounce (v) to abolish, usually by authority
2  abrogate (v) to abolish, usually by authority
3  acerbic (adj) biting, bitter in tone or taste
4  acrimony (n) bitterness, discord
5  acumen (n) keen insight
6  adumbrate (v) to sketch out in a vague way
7  alacrity (n) eagerness, speed
8  anathema (n) a cursed, detested person
9  antipathy (n) a strong dislike, repugnance
10 approbation (n) praise
11 arrogate (v) to take without justification
12 ascetic (adj) practicing restraint/self-discipline, usually religious
13 aspersion (n) a curse, expression of ill-will
14 assiduous (adj) hard-working, diligent
15 blandish (v) to coax by using flattery
16 boon (n) a gift or blessing
17 brusque (adj) short, abrupt, dismissive
18 buffet (v) to strike with force;(n) an arrangement of food
19 burnish (v) to polish, shine
20 buttress (v) to support, hold up;(n) something that offers support
21 cacophony (n.) tremendous noise, disharmonious sound
22 cajole (v.) to urge, coax
23 calumny (n.) attempt to spoil someone else\'s reputation/spreading lies
24 capricious (adj.) subject to whim, fickle
25 clemency (n.) mercy
26 cogent (adj.) intellectually convincing
27 concomitant (adj.) accompanying in a subordinate fashion
28 conflagration (n.) great fire
29 contrite (adj.) penitent, eager to be forgiven
30 conundrum (n.) puzzle, problem
31 credulity (n.) readiness to believe
32 cupidity (n.) greed, strong desire
33 cursory (adj.) brief to the point of being superficial
34 decry (v.) to criticize openly
35 defile (v.) to make unclean, impure
36 deleterious (adj.) harmful
37 demure (adj.) quiet, modest, reserved
```

The program will read the first word of each line of the file and use them as the possible words for the Hangman game. At the end of the game, the program will display the rest of the line (i.e.: the definition of the word) that the hidden word was chosen from. This shows up at the bottom of the panel.

You can make up a list of words, keeping in mind that the program will read the first word in each line of your file as a candidate hidden word, and will display the rest of the line at the end of the game.

```
package hangmannew;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class HangmanNew extends JFrame {

    private static String used_string = "";
    private static String state_string = "";
    private static String state_display = "";
    private static String choice = " ";
    private static JLabel used;
    private static JLabel state;
    private static boolean win = false;
    private static int random_index;
    private static final int max_guesses = 13;
    private static Counter counter1 = new Counter("counter1", 1000);
    private static String hidden_word = " ";
    private static String hidden_definition;
    private static final int n = 30; // number of letters
    private static JButton[] letters = new JButton[n];
    private static String[] fileItems = new String[]{"New", "Quit"};
    private static char[] fileShortcuts = {'N', 'Q'};

    public HangmanNew() throws FileNotFoundException {
        //set the secret word, randomly chosen from a file
        setWord();

        //set up this JFrame's menu bar and options:
        JMenuBar myMenuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        for (int i = 0; i < fileItems.length; i++) {
```

```
            JMenuItem item = new JMenuItem(fileItems[i],
fileShortcuts[i]);
            item.addActionListener(new menuListener());
            fileMenu.add(item);
        }
        myMenuBar.add(fileMenu);
        this.setJMenuBar(myMenuBar);

        //Three JPanels added to the JFrame
        Drawing drawing = new Drawing(); //hangman gallows drawing
        LetterButtons buttons = new LetterButtons();//letters to
choose from
        LetterLabels labels = new LetterLabels();//ongoing results

        add(drawing, BorderLayout.CENTER);
        add(buttons, BorderLayout.EAST);
        add(labels, BorderLayout.SOUTH);

    }

    public static void setWord() throws FileNotFoundException {
        //The text file "most_difficult_words_defined.txt" must be
in the project directory
        //In this case it contains the 250 most commonly missed SAT
words.
        java.io.File words_file = new
java.io.File("most_difficult_words_defined.txt");
        Scanner input = new Scanner(words_file);
        ArrayList<String> defined_list = new ArrayList<String>();
        while (input.hasNextLine()) {
            String line = input.nextLine();
            defined_list.add(line);
        }
        random_index = (int) (defined_list.size() * Math.random());
        hidden_definition = defined_list.get(random_index);
        String parts[] = hidden_definition.split(" ");
        hidden_word = parts[0];
    }

    public static void restart() throws FileNotFoundException {
        win = false;
        setWord();
        used_string = "";
        state_string = "";
        state_display = "";
        for (int i = 0; i < hidden_word.length(); i++) {
            state_string += " ";
            state_display += "_ ";
        }
        state.setText(state_display);
        used.setText(" ");
        counter1.setValue(0);
        choice = " ";
        for (int i = 0; i < n; i++) {
```

```
            letters[i].setBackground(Color.white);
            letters[i].setForeground(Color.black);
        }
    }

    class LetterButtons extends JPanel {

        public LetterButtons() {

            setLayout(new GridLayout(5, 6, 0, 0));
            for (int i = 0; i < n; i++) {
                letters[i] = new JButton("" + (char) (i + 65));
                add(letters[i]);
                letters[i].addActionListener(new buttonListener());
            }
        }
    }

    class LetterLabels extends JPanel {

        LetterLabels() {

            for (int i = 0; i < hidden_word.length(); i++) {
                state_string += " ";
                state_display += "_ ";
            }
            state = new JLabel(state_display.toUpperCase());
            Font font1 = new Font("Courier", Font.BOLD, 36);
            state.setForeground(Color.black);
            state.setFont(font1);
            state.setHorizontalAlignment(JLabel.CENTER);

            used = new JLabel(state_string);
            used.setForeground(Color.black);
            Font font2 = new Font("Courier", Font.BOLD, 26);
            used.setFont(font2);
            used.setHorizontalAlignment(JLabel.CENTER);

            this.setLayout(new BorderLayout());
            add(state, BorderLayout.NORTH);
            add(used, BorderLayout.SOUTH);

        }
    }

    class Drawing extends JPanel {

        protected void paintComponent(Graphics g) {

            super.paintComponent(g);
            Image image = new ImageIcon("images/image" +
counter1.value() + ".jpg").getImage();
            g.drawImage(image, 0, 0, image.getHeight(null),
image.getHeight(null), this);
```

```
            if (counter1.value() >= 13 && !win) {
                Image imageRIP = new
ImageIcon("images/RIP.jpg").getImage();
                int width = getWidth();
                int height = getHeight();
                g.drawImage(imageRIP, 0, 0, width, height, this);
            }

            Image imageWin = new
ImageIcon("images/HappyStickman2.jpg").getImage();
            if (win) {
                g.drawImage(imageWin, 0, 0, getWidth(), getHeight(),
this);
            }
        }
    }

    class menuListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println(" action command = " +
e.getActionCommand());
            if (e.getActionCommand().equals("Quit")) {
                System.exit(0);
            }
            if (e.getActionCommand().equals("New")) {
                try {
                    restart();
                } catch (FileNotFoundException ex) {

Logger.getLogger(HangmanNew.class.getName()).log(Level.SEVERE, null,
ex);
                }
            }
        }
    }

    class buttonListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            choice = e.getActionCommand();
            for (int i = 0; i < n; i++) {
                if
(letters[i].getActionCommand().equals(e.getActionCommand())) {
                    letters[i].setForeground(Color.RED);
                }
            }

            if (used_string.indexOf(choice) < 0) {
                used_string += choice;
            } else {
```

```
                JOptionPane.showMessageDialog(null, "Letter " +
choice + " is already used");
            }

            choice = choice.toLowerCase();
            char[] char_choice = choice.toCharArray();
            char[] char_hidden = hidden_word.toCharArray();
            char[] char_state = state_string.toCharArray();
            boolean hit = false;
            for (int i = 0; i < char_hidden.length; i++) {
                if (char_hidden[i] == char_choice[0]) {
                    char_state[i] = char_hidden[i];
                    hit = true;
                }
            }
            if (!hit) {
                counter1.increment();
            }
            state_string = "";
            state_display = "";
            for (int i = 0; i < char_state.length; i++) {
                state_string += char_state[i];
                if (char_state[i] == ' ') {
                    state_display += "_ ";
                } else {
                    state_display += char_state[i] + "";
                }
            }

            state.setText(state_display.toUpperCase());
            if (counter1.value() >= max_guesses) {
                used.setForeground(Color.white);
                used.setOpaque(true);
                used.setBackground(Color.gray);
                Font font3 = new Font("Courier", Font.BOLD, 18);
                used.setFont(font3);
                used.setText(" Sorry, You Lose! (But, keep
playing!)");
            }
            int hit_count = 0;
            for (int i = 0; i < char_hidden.length; i++) {
                if (char_hidden[i] == char_state[i]) {
                    hit_count++;
                }
            }

            if (hit_count == hidden_word.length()) {
                win = true;
                used.setForeground(Color.black);
                used.setOpaque(true);
                used.setBackground(Color.white);
                Font font3 = new Font("Courier", Font.BOLD, 18);
                used.setFont(font3);
                used.setText(" " + hidden_definition + " ");
```

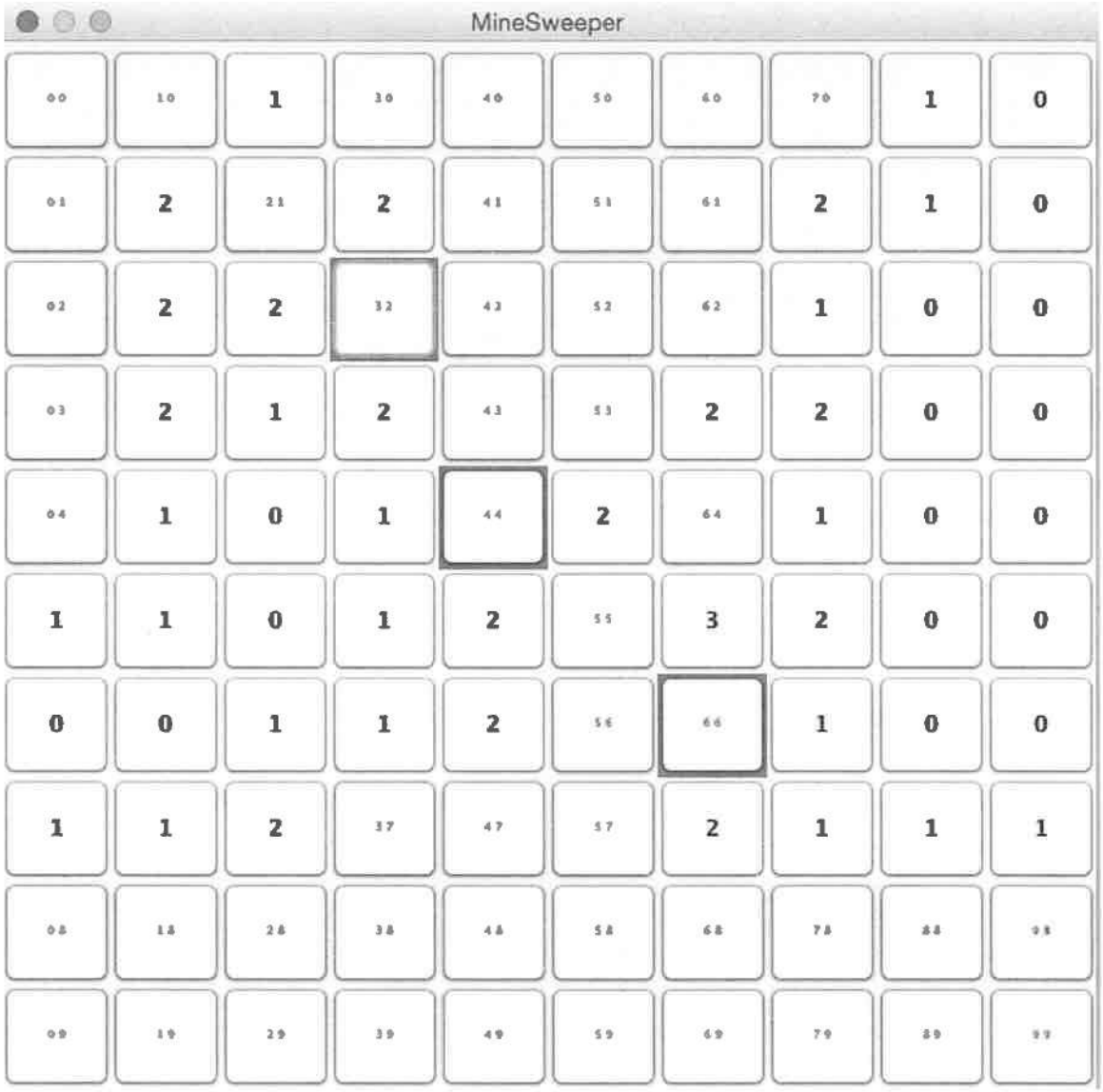
```
            }
            repaint();
        }
    }

    public static void main(String[] args) throws
FileNotFoundException {
        HangmanNew hangman = new HangmanNew();
        hangman.setTitle("HANGMAN");
        hangman.setSize(new Dimension(700, 400));
        hangman.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        hangman.setLocationRelativeTo(null);
        hangman.setVisible(true);

    }
}

////////////////////////////////////
package hangmannew;

public class Counter implements Comparable<Counter> {

    private final String name;     // counter name
    private final int maxCount;    // maximum value
    private int count;             // current value

    // create a new counter with the given parameters
    public Counter(String id, int max) {
        name = id;
        maxCount = max;
        count = 0;
    }

    // increment the counter by 1
    public void increment() {
        if (count < maxCount) count++;
        assert count >= 1;
    }

    // return the current count
    public int value() {
        return count;
    }

    // return a string representation of this counter
    public String toString() {
        return name + ": " + count;
    }

    // compare two Counter objects based on their count
    public int compareTo(Counter that) {
        if      (this.count < that.count) return -1;
        else if (this.count > that.count) return +1;
        else                              return  0;
```

```
    }

    public void setValue(int n){
        count = n;
    }

}
```

Java Project: Minesweeper

Description: Play Minesweeper. Left-click to open a cell. Right-click to mark the cell a bomb. Find all the bombs and win.

Sample Output:



```
package minesweeper;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Scanner;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Minesweeper extends JFrame {
    final int N = 10;//NxN grid of cells 100
    int[][] grid = new int[N][N];
    int[][] cover = new int[N][N];
    JButton[][] buttons = new JButton[N][N];
    int colClick;
    int rowClick;
    boolean game_over = false;
    Font f2 = new Font("New Courier", Font.BOLD, 12);
    int bombs_marked = 0;
    int bombs_actual;

    public Minesweeper() {
        initializeButtons();
        initializeBombs();
        initializeNbhds();
    }
public void initializeButtons(){
    JPanel panel = new JPanel(new GridLayout(N, N));
    Font f1 = new Font("New Courier", Font.PLAIN, 6);
    for (int j = 0; j < N; j++) {
        for (int i = 0; i < N; i++) {
            buttons[i][j] = new JButton();
            buttons[i][j].setFont(f1);
            buttons[i][j].setText(i + " " + j);
            buttons[i][j].setVisible(true);
            buttons[i][j].addActionListener(new buttonListener());
            buttons[i][j].addMouseListener(new myMouseListener(i,
j));
            buttons[i][j].setOpaque(true);
            panel.add(buttons[i][j]);
        }
    }
    add(panel);
}

public void initializeNbhds() {
    for (int j = 0; j < N; j++) {
```

```
        for (int i = 0; i < N; i++) {
            if (grid[i][j] != -1) {
                grid[i][j] = getSum(i, j);
            }
        }
    }
}
public int getSum(int i, int j) {
    int sum = 0;
    for (int q = j - 1; q <= j + 1; q++) {
        for (int p = i - 1; p <= i + 1; p++) {
            if (p >= 0 && p< N && q >= 0 && q < N) {
                if (grid[p][q] == -1) {
                    sum++;
                }
            }
        }
    }
    return sum;
}
public void initializeBombs() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            int number = (int) ( 0.5 * N * Math.random());
            System.out.println("number = " + number);
            if (number == 0) {
                grid[i][j] = -1;
                bombs_actual++;
            } else{
                grid[i][j] = 0;
            }
            cover[i][j] = 0;
        }
    }
}

class buttonListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {

        Scanner scanner = new Scanner(e.getActionCommand());
        if (scanner.hasNextInt());{
            colClick = scanner.nextInt();
        }
        if (scanner.hasNextInt()) {
            rowClick = scanner.nextInt();
        }
        if (grid[colClick][rowClick] == -1) {
            game_over = true;
            JOptionPane.showMessageDialog(null, "KaBoom -- game
over");
        } else {
            cover[colClick][rowClick] = 1;
```

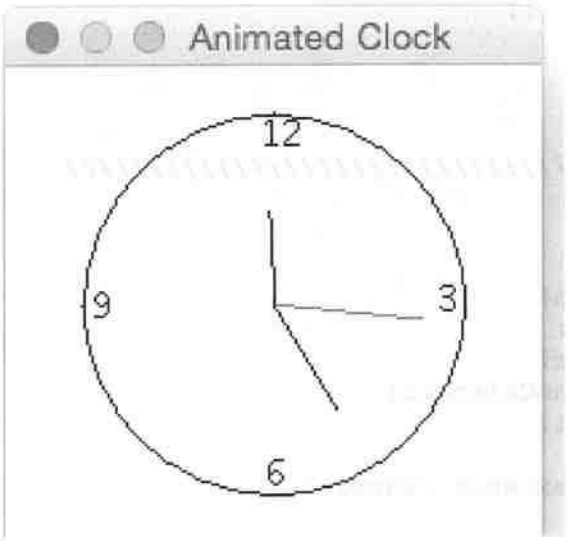
```
            buttons[colClick][rowClick].setFont(f2);
            buttons[colClick][rowClick].setBackground(Color.white);
            buttons[colClick][rowClick].setText("" +
grid[colClick][rowClick]);
            if (grid[colClick][rowClick] == 0) {
                uncoverNbhd(colClick, rowClick);
            }
        }
    }
    public void uncoverNbhd(int i, int j) {
        for (int q = j - 1; q <= j + 1; q++) {
            for (int p = i - 1; p <= i + 1; p++) {
                if (p >= 0 && p < N && q >= 0 && q < N) {
                    if (cover[p][q] ==0) {
                        if (grid[p][q] != -1) {
                            cover[p][q] = 1;
                            buttons[p][q].setFont(f2);

buttons[p][q].setBackground(Color.white);
                            buttons[p][q].setText("" + grid[p][q]);
                        }
                        if (grid[p][q] == 0) {
                            uncoverNbhd(p, q);
                        }
                    }
                }
            }
        }
    }
}

class myMouseListener extends MouseAdapter {

    private int i, j;

    public myMouseListener(int i, int j) {
        this.i = i;
        this.j = j;
    }

    @Override
    public void mouseClicked(MouseEvent e) {
        if(e.getButton() == e.BUTTON3) {
            Color this_color = new Color(238, 238, 238);
            if (buttons[i][j].getBackground().equals(this_color)) {
                buttons[i][j].setBackground(Color.red);
            } else {
                buttons[i][j].setBackground(this_color);
            }
            bombs_marked = 0;
            for (int ii = 0; ii < N; ii++) {
                for (int jj = 0; jj < N; jj++)
                    if
(buttons[ii][jj].getBackground().equals(Color.red)) {
```

```
                        if (grid[ii][jj] == -1) {
                            bombs_marked++;
                        } else {
                            bombs_marked--;
                        }
                    }
                }
            }
            System.out.println("bombs_marked = " + bombs_marked);
            System.out.println("bombs_actual = " + bombs_actual);
            if (bombs_marked == bombs_actual) {
                JOptionPane.showMessageDialog(null, "You Win!!");
            }
        }
    }

public static void main(String[] args) {
    Minesweeper minesweeper = new Minesweeper();
    minesweeper.setSize(new Dimension(600, 600));
    minesweeper.setTitle("MineSweeper");
    minesweeper.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    minesweeper.setLocationRelativeTo(null);
    minesweeper.setVisible(true);}
}
```

Java Project: ClockAnimation

Description: Shows a simulated clock, with hour, minute and second hand, with accurate time. It uses a StillClock.java class in addition to the main class ClockAnimation.java

Sample Output:



There are two java files: ClockAnimation.java and StillClock.java. The program could use some embellishments, such as a better drawing of the clock and better/more numbers etc.

```
package clockanimation;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.Timer;

public class ClockAnimation extends JFrame {

    private StillClock clock = new StillClock();

    public ClockAnimation() {
        this.add(clock);
        Timer timer = new Timer(1000, new TimerListener());
        timer.start();
    }
```

```
    public static void main(String[] args) {
        JFrame frame = new ClockAnimation();
        frame.setTitle("Animated Clock");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //frame.setLocationRelativeTo(null);
        frame.setSize(200, 200);
        frame.setVisible(true);
    }

    public class TimerListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            clock.setCurrentTime();
            clock.repaint();
        }
    }
}

//////////////////////////////////////////////////////
package clockanimation;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.util.Calendar;
import java.util.GregorianCalendar;
import javax.swing.JPanel;

public class StillClock extends JPanel {

    private int hour;
    private int minute;
    private int second;

    public StillClock() {
        setCurrentTime();
    }

    public StillClock(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }

    public int getHour() {
        return hour;
    }

    public int getMinute() {
        return minute;
    }

    public int getSecond() {
        return second;
```

```
    }

    public void setHour(int hour) {
        this.hour = hour;
    }

    public void setMinute(int minute) {
        this.minute = minute;
    }

    public void setSecond(int second) {
        this.second = second;
    }

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int clockRadius = (int) (Math.min(getWidth(), getHeight()) *
0.8 * 0.5);
        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;

        g.setColor(Color.black);
        g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
                2 * clockRadius, 2 * clockRadius);
        g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
        g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
        g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
        g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);

        //Draw second hand:
        //Note: it may appear that the sin and cos should be
reversed,
        //but careful analysis shows we are dealing with cos(pi/2 -
theta)
        //which is equal to sin(theta), so the sin and cos get
exchanged.
        int sLength = (int) (clockRadius * 0.8);
        int xSecond = (int) (xCenter + sLength * Math.sin(second * 2
* Math.PI / 60));
        int ySecond = (int) (yCenter - sLength * Math.cos(second * 2
* Math.PI / 60));
        g.setColor(Color.red);
        g.drawLine(xCenter, yCenter, xSecond, ySecond);

        //Draw minute hand:
        int mLength = (int) (clockRadius * 0.65);
        int xMinute = (int) (xCenter + mLength * Math.sin(minute * 2
* Math.PI / 60));
        int yMinute = (int) (yCenter - mLength * Math.cos(minute * 2
* Math.PI / 60));
        g.setColor(Color.black);
        g.drawLine(xCenter, yCenter, xMinute, yMinute);
```

```
        //Draw hour hand:
        int hLength = (int) (clockRadius * 0.5);
        int xHour = (int) (xCenter + hLength * Math.sin(hour * 2 *
Math.PI / 12));
        int yHour = (int) (yCenter - hLength * Math.cos(hour * 2 *
Math.PI / 12));
        g.setColor(Color.black);
        g.drawLine(xCenter, yCenter, xHour, yHour);
    }

    public void setCurrentTime() {
        Calendar calendar = new GregorianCalendar();
        this.hour = calendar.get(Calendar.HOUR_OF_DAY);
        this.minute = calendar.get(Calendar.MINUTE);
        this.second = calendar.get(Calendar.SECOND);
    }

    public Dimension getPreferredSize() {
        return new Dimension(400, 400);
    }
}
```

Java Project: SimpleAnimation

Description: This program shows a single ball (randomly generated : size, direction and color) bouncing in the drawing panel.

Sample Output:


```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package simpleanimation;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;
/**
 *
 * @author Adrian
 */
public class SimpleAnimation extends JFrame {

    public static final int W = 600;
    public static final int H = 600;
    public static double radius;
```

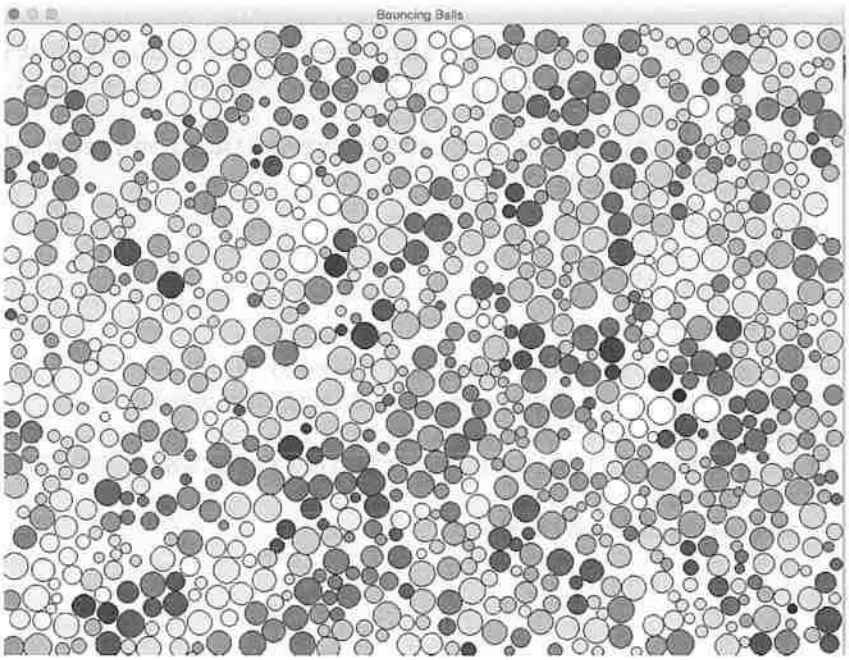
```
    public static double maxRadius = W / 2;
    public static double dx;
    public static double dy;
    public static double x;
    public static double y;
    public static Color color;
    public static double maxSpeed = 20;

    public SimpleAnimation() {
        MovingBallPanel panel = new MovingBallPanel();
        this.add(panel);
        initializeBall();
    }
    private void initializeBall(){
        Random rand = new Random();
        //create a radius between 0 and max
        radius = maxRadius * Math.random();
        //start the ball in the center of window
        x = W * Math.random();
        y = H * Math.random();
        color = new Color(rand.nextInt(256), rand.nextInt(256),
rand.nextInt(256));
        //start ball with random dx and dy between +maxspeed/2 and -
maxspeed/2
        dx = maxSpeed * Math.random() - maxSpeed / 2;
        dy = maxSpeed * Math.random() - maxSpeed / 2;
    }
    static class MovingBallPanel extends JPanel {
        //sends an event ever 30 milliseconds
        //timer listener reacts to each
        //of these events by calling the method repaint();
        private Timer timer = new Timer(30, new TimerListener());

        public MovingBallPanel(){
            timer.start();
        }

        //must be called in order to draw to any JPanel
        //Graphics object g is from the OS
        protected void paintComponent(Graphics g){
            super.paintComponent(g);

            checkBoundary();
            updateXYPosition();
            drawBall(g);
        }
            private void checkBoundary() {
                if(x + radius > getWidth()){
                    dx = -Math.abs(dx);
                }
                if (x - radius < 0){
                    dx = Math.abs(dx);
                }
                if( y + radius > getHeight()){
```

```
                dy = -Math.abs(dy);
            }
            if(y - radius < 0){
                dy = Math.abs(dy);
            }
        }
        private void updateXYPosition(){
            x =x + dx;
            y = y + dy;
        }
        private void drawBall(Graphics g){
            g.setColor(color);
            g.fillOval((int) (x - radius),(int) (y - radius) ,
(int) (2 * radius),(int) (2 * radius));
            g.setColor(Color.black);
            g.drawOval((int) (x - radius),(int) (y - radius) ,
(int) (2 * radius),(int) (2 * radius));
        }
        class TimerListener implements ActionListener{
             //this metho, action performed is called at each
event
            public void actionPerformed(ActionEvent e){
                repaint();
            }
        }
    }


    public static void main(String[] args) {
        SimpleAnimation frame = new SimpleAnimation();
        frame.setTitle("Animated Ball");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W,H);
        frame.setVisible(true);

    }
}
```

Java Project: AnimatedBalls

Description: This program shows many balls bouncing in the ddrawing panel. You can change the number of balls, their size, speed and color. You can also change what happens when they collide. The default code just has them bounce off each other and change color in a certain way (look at the code to see how/when a ball's color changes), but you can invent other things happens such as explosions etc.

Sample Output:



```
package animatedballs;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Random;
import java.util.Vector;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

public class AnimatedBalls extends JFrame {

    public static int W = 900;
    public static int H = 700;
    public static int n = 1000;
    public static int m = 6;
    public static double[] radius = new double[n];
    public static double minRadius = 5;
    public static double maxRadius = 15;
    public static double[] dx = new double[n];
    public static double[] dy = new double[n];
    public static double[] x = new double[n];
    public static double[] y = new double[n];
    public static Color[] color = new Color[n];
    public static double speed = 1;
    public static int[] active = new int[n];
    public static int[] score = new int[n];
    public static int[][] gene = new int[n][m];

    public AnimatedBalls() {
        MovingBallPanel panel = new MovingBallPanel();
        this.add(panel);
        initializeBalls();
        printBalls();

    }

    public static void printBalls() {
        for (int i = 0; i < n; i++) {
            System.out.print(i);
            System.out.print(" ");
            for (int g = 0; g < m; g++) {
                System.out.print(gene[i][g]);
            }
            System.out.println("");
        }
    }

    private void initializeBalls() {
        Random rand = new Random();
```

```
        int r = 0;
        int g = 0;
        int b = 0;
        for (int i = 0; i < n; i++) {
            radius[i] = minRadius + (maxRadius - minRadius) *
Math.random();

            x[i] = radius[i] + (W - 2 * radius[i]) *
rand.nextDouble();
            y[i] = radius[i] + (H - 2 * radius[i]) *
rand.nextDouble();

            //x[i] = W/2;
            //y[i] = H/2;
            //color[i] = new Color(rand.nextInt(256),
rand.nextInt(256), rand.nextInt(256));
            active[i] = 1;
            for (int k = 0; k < m; k++) {
                gene[i][k] = (int) (2 * Math.random());
            }
            r = setColor(i, gene[i][0], gene[i][1]);
            g = setColor(i, gene[i][2], gene[i][3]);
            b = setColor(i, gene[i][4], gene[i][5]);
            color[i] = new Color(r, g, b);
            score[i] = setScore(i);

            dx[i] = score[i] * Math.random() - score[i] / 2;
            dy[i] = score[i] * Math.random() - score[i] / 2;

        }
        //color[n-1] = Color.GRAY;
    }

    public static int setScore(int i) {
        int sum = 0;
        for (int k = 0; k < m; k++) {
            sum += gene[i][k];
        }
        return sum;
    }

    public static int setColor(int i, int v1, int v2) {
        if (v1 == 0 && v2 == 0) {
            return (int) (64 * Math.random());
            //return 0;
        } else if (v1 == 0 && v2 == 1) {
            return 64 + (int) (64 * Math.random());
        } else if (v1 == 1 && v2 == 0) {
            return 128 + (int) (64 * Math.random());
        } else if (v1 == 1 && v2 == 1) {
            return 192 + (int) (64 * Math.random());
            //return 255;
        }
        return 0;
```

```
    }

    static class MovingBallPanel extends JPanel {

        private Timer timer = new Timer(30, new TimerListener());
        public boolean timer_stop = false;

        public MovingBallPanel() {
            timer.start();
            this.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    int delay = timer.getDelay();
                    if (e.getButton() == MouseEvent.BUTTON1) {
                        if (timer_stop) {
                            timer.start();
                            timer_stop = false;
                        } else {
                            timer.stop();
                            timer_stop = true;
                        }

                        //timer.setDelay(delay > 5 ? delay - 5 : 5);
                    } else if (e.getButton() == MouseEvent.BUTTON3)
{
                        timer.setDelay(delay < 5000 ? delay + 5 :
5000);
                    }
                }

            });
        }

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            checkBoundary();
            checkCollisions();
            updateXYPositions();
            //pumpVelocities();
            drawBalls(g);
            //printBalls();
        }

        private void checkBoundary() {
            for (int i = 0; i < n; i++) {
                if (x[i] + radius[i] > getWidth()) {
                    dx[i] = -Math.abs(dx[i]);
                }

                if (x[i] - radius[i] < 0) {
                    dx[i] = Math.abs(dx[i]);
                }

                if (y[i] + radius[i] > getHeight()) {
                    dy[i] = -Math.abs(dy[i]);
```

```
                }

                if (y[i] - radius[i] < 0) {
                    dy[i] = Math.abs(dy[i]);
                }
            }
        }

        private void updateXYPositions() {
            for (int i = 0; i < n; i++) {
                x[i] = x[i] + dx[i];
                y[i] = y[i] + dy[i];
            }
        }

        private void drawBalls(Graphics g) {
            for (int i = 0; i < n; i++) {
                if (active[i] == 1) {

                    g.setColor(color[i]);
                    g.fillOval((int) (x[i] - radius[i]), (int) (y[i]
- radius[i]), (int) (2 * radius[i]), (int) (2 * radius[i]));
                    g.setColor(Color.black);
                    g.drawOval((int) (x[i] - radius[i]), (int) (y[i]
- radius[i]), (int) (2 * radius[i]), (int) (2 * radius[i]));
                    //g.drawString("" + i, (int) (x[i]), (int)
(y[i]));
                }
            }
        }

        private void checkCollisions() {
            for (int i = 0; i < n; i++) {
                for (int j = i + 1; j < n; j++) {
                    double dist = Math.sqrt(Math.pow(x[j] - x[i], 2)
+ Math.pow(y[j] - y[i], 2));

                    if (active[i] == 1 && active[j] == 1) {
                        if (dist > 0 && dist <= (radius[i] +
radius[j])) {
                            if (score[i] > score[j]) {
                                /*for (int k = 0; k < m; k++) {
                                    gene[j][k] = gene[i][k];
                                }
                                int r = setColor(j, gene[j][0],
gene[j][1]);
                                int g = setColor(j, gene[j][2],
gene[j][3]);
                                int b = setColor(j, gene[j][4],
gene[j][5]);
                                color[j] = new Color(r, g, b);
                                score[j] = setScore(j);
                                 */
                                color[j] = color[i];
```

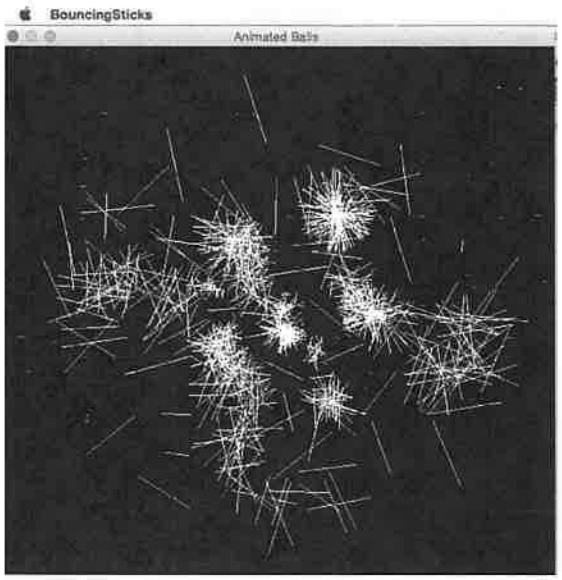
```
                            }
                            /*
                            else {
                                for (int k = 0; k < m; k++) {
                                    gene[i][k] = gene[j][k];
                                }
                                int r = setColor(i, gene[i][0],
gene[i][1]);
                                int g = setColor(i, gene[i][2],
gene[i][3]);
                                int b = setColor(i, gene[i][4],
gene[i][5]);
                                color[i] = new Color(r, g, b);
                                score[i] = setScore(i);
                                //color[i] = color[j];
                            }
                              */

                            elasticCollision(i, j);

                        }


                    }
                }
            }
        }

        public void pumpVelocities() {
            double epsilon = 0.5;
            for (int i = 0; i < n; i++) {
                if (Math.abs(dx[i]) < epsilon) {
                    dx[i] *= 1.0;
                }
                if (Math.abs(dy[i]) < epsilon) {
                    dy[i] *= 1.0;
                }
            }
        }

        private void checkCollissions2() {

        }

        public void elasticCollision(int i, int j) {
            // get the mtd

            double delta_x = x[i] - x[j];
            double delta_y = y[i] - y[j];
            double d = Math.sqrt((delta_x * delta_x + delta_y *
delta_y));

            double mtd_x = ((radius[i] + radius[j] - d) / d) *
delta_x;
```

```
            double mtd_y = ((radius[i] + radius[j] - d) / d) *
delta_y;
            double mtd_length = Math.sqrt(mtd_x * mtd_x + mtd_y *
mtd_y);
            double mtd_norm_x = mtd_x / mtd_length;
            double mtd_norm_y = mtd_y / mtd_length;

            double im1 = 1.0 / radius[i];
            double im2 = 1.0 / radius[j];

            x[i] += (im1 / (im1 + im2)) * mtd_x;
            y[i] += (im1 / (im1 + im2)) * mtd_y;

            x[j] += -(im2 / (im1 + im2)) * mtd_x;
            y[j] += -(im2 / (im1 + im2)) * mtd_y;
            ////////////////////////////////////
            double v_x = dx[i] - dx[j];
            double v_y = dy[i] - dy[j];

            double vn = v_x * mtd_norm_x + v_y * mtd_norm_y;

            if (vn > 0.0) {
                return;
            }
            double elastic_constant = 1.0;

            double impulse_x = mtd_norm_x * (-(1 + elastic_constant)
* vn) / (im1 + im2);
            double impulse_y = mtd_norm_y * (-(1 + elastic_constant)
* vn) / (im1 + im2);

            dx[i] += im1 * impulse_x;
            dy[i] += im1 * impulse_y;

            dx[j] += -im2 * impulse_x;
            dy[j] += -im2 * impulse_y;

        }

        class TimerListener implements ActionListener {

            public void actionPerformed(ActionEvent e) {
                repaint();

            }
        }
    }

    public static void main(String[] args) {
        AnimatedBalls frame = new AnimatedBalls();
        frame.setTitle("Bouncing Balls");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W, H);
        frame.setVisible(true);
```

Java Project: BouncingSticks

Description: Displays randomly generating rotating sticks.

Sample Output:




```
package bouncingsticks;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Random;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

public class BouncingSticks extends JFrame {

    public static int W = 600;
    public static int H = 600;

    public BouncingSticks() {
        MovingBallPanel m1 = new MovingBallPanel();
        this.add(m1);
    }

    public static void main(String[] args) {
```

```
        BouncingSticks frame = new BouncingSticks();
        frame.setTitle("Animated Balls");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W, H);
        frame.setVisible(true);
    }

    static class MovingBallPanel extends JPanel {

        public static int n = 800;
        public static double[] a = new double[n];

        public static double[] da = new double[n];
        public static double[] r = new double[n];
        public static double[] dx = new double[n];
        public static double[] dy = new double[n];
        public static double[] x = new double[n];
        public static double[] y = new double[n];
        public static Color[] color = new Color[n];
        private Timer timer = new Timer(30, new TimerListener());

        public MovingBallPanel() {

            this.setBackground(Color.black);
            initializeBalls();
            timer.start();
            this.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    int delay = timer.getDelay();
                    if (e.getButton() == MouseEvent.BUTTON1) {
                        timer.setDelay(delay > 5 ? delay - 5 : 5);
                    } else if (e.getButton() == MouseEvent.BUTTON3)
{
                        timer.setDelay(delay < 5000 ? delay + 5 :
5000);
                    }
                }
            });
        }

        private void initializeBalls() {
            Random rand = new Random();
            double maxBallRadius = 40;
            double minBallRadius = 2;
            double maxBallSpeed = 10;
            double maxRotSpeed = 1.0;
            for (int i = 0; i < n; i++) {
                r[i] = minBallRadius + (maxBallRadius *
Math.random());
                x[i] = W / 2;
                y[i] = H / 2;
                int red = rand.nextInt(256);
                int green = rand.nextInt(256);
                int blue = rand.nextInt(256);
```

```
                color[i] = new Color(red, green, blue);
                color[i] = new Color(255,255,255);
                dx[i] = maxBallSpeed * Math.random() - maxBallSpeed
/ 2;
                dy[i] = maxBallSpeed * Math.random() - maxBallSpeed
/ 2;
                a[i] = 2.0 * Math.PI * Math.random();
                da[i] = maxRotSpeed * Math.random() - maxRotSpeed /
2;

            }
        }

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);

            checkBoundary();//check ball at window boundary
            checkCollisions();//check for collisions between balls
            checkZeroVelocity(); // want ALL the balls to move.
            updateXYPositions();//use dx, dy to get new x, y
            drawBalls(g);//draw all the balls
        }

        private void checkBoundary() {
            for (int i = 0; i < n; i++) {
                if (x[i] + r[i] > getWidth()) {
                    dx[i] = -Math.abs(dx[i]);
                }
                if (x[i] - r[i] < 0) {
                    dx[i] = Math.abs(dx[i]);
                }
                if (y[i] + r[i] > getHeight()) {
                    dy[i] = -Math.abs(dy[i]);
                }
                if (y[i] - r[i] < 0) {
                    dy[i] = Math.abs(dy[i]);
                }
            }
        }

        private void updateXYPositions() {
            for (int i = 0; i < n; i++) {
                x[i] = x[i] + dx[i];
                y[i] = y[i] + dy[i];
                a[i] = a[i] + da[i];
            }
        }

        private void drawBalls(Graphics g) {
            for (int i = 0; i < n; i++) {
                g.setColor(color[i]);
                g.drawLine(
                        (int) (x[i] + r[i] * Math.cos(a[i] +
Math.PI)),
```

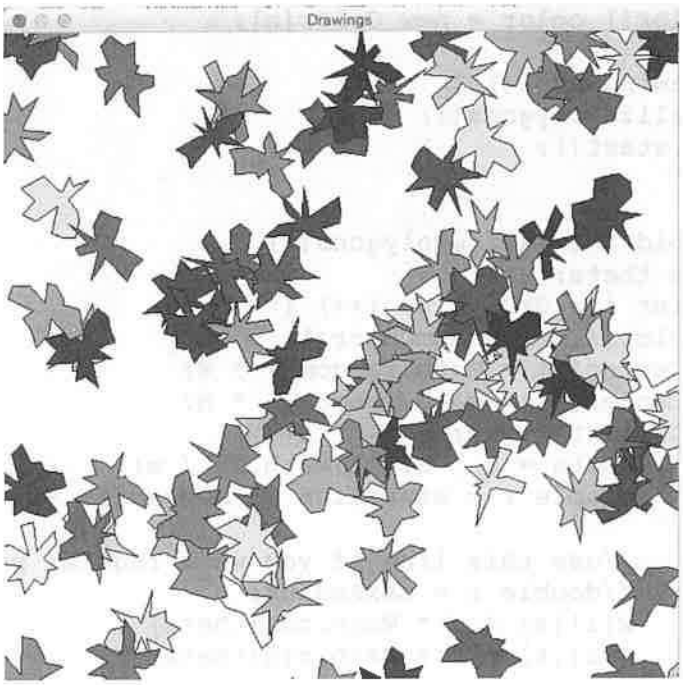
```
                        (int) (y[i] + r[i] * Math.sin(a[i] +
Math.PI)),
                        (int) (x[i] + r[i] * Math.cos(a[i])),
                        (int) (y[i] + r[i] * Math.sin(a[i])));
            }
        }

        private void checkCollisions() {
            double[] unit = new double[2];
            double length = 0.0;
            double iLength = 0.0;
            double jLength = 0.0;
            double temp = 0.0;
            for (int i = 0; i < n; i++) {
                for (int j = i + 1; j < n; j++) {
                    //check to see if ball i has collided with ball
j:
                    double dist = Math.sqrt(
                            Math.pow(x[j] - x[i], 2)
                            + Math.pow(y[j] - y[i], 2));
                    if (dist > 0 && dist <= r[i] + r[j]) {
                        unit[0] = (x[j] - x[i]) / dist;
                        unit[1] = (y[j] - y[i]) / dist;
                        length = Math.sqrt(
                                Math.pow(unit[0], 2)
                                + Math.pow(unit[1], 2));

                        iLength = Math.sqrt(
                                Math.pow(dx[i], 2)
                                + Math.pow(dy[i], 2));
                        jLength = Math.sqrt(
                                Math.pow(dx[j], 2)
                                + Math.pow(dy[j], 2));

                        dx[i] = -jLength * unit[0];
                        dy[i] = -jLength * unit[1];

                        dx[j] = iLength * unit[0];
                        dy[j] = iLength * unit[1];

                        temp = a[i];
                        a[i] = a[j];
                        a[j] = temp;
                    }
                }
            }
        }

        private void checkZeroVelocity() {
            for (int i = 0; i < n; i++) {
                if ((dx[i] * dx[i] + dy[i] * dy[i]) <= 0.1) {
                    System.out.println("zero vel at " + i);
                    dx[i] = 1 + (4 * Math.random() - 2);
                    dy[i] = 1 + (4 * Math.random() - 2);
```

```
                }
            }
        }

        class TimerListener implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                repaint();
            }
        }
    }
}
```

Java Project: RotatingPolygon

Description: Draw randomly placed polygons, and have them rotate in place.

Sample Output:




```
package rotatingpolygon;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

public class RotatingPolygon extends JFrame {

    public static int W = 600;
    public static int H = 600;

    public RotatingPolygon() {
        DrawingPanel drawingPanel = new DrawingPanel();
        this.add(drawingPanel);
```

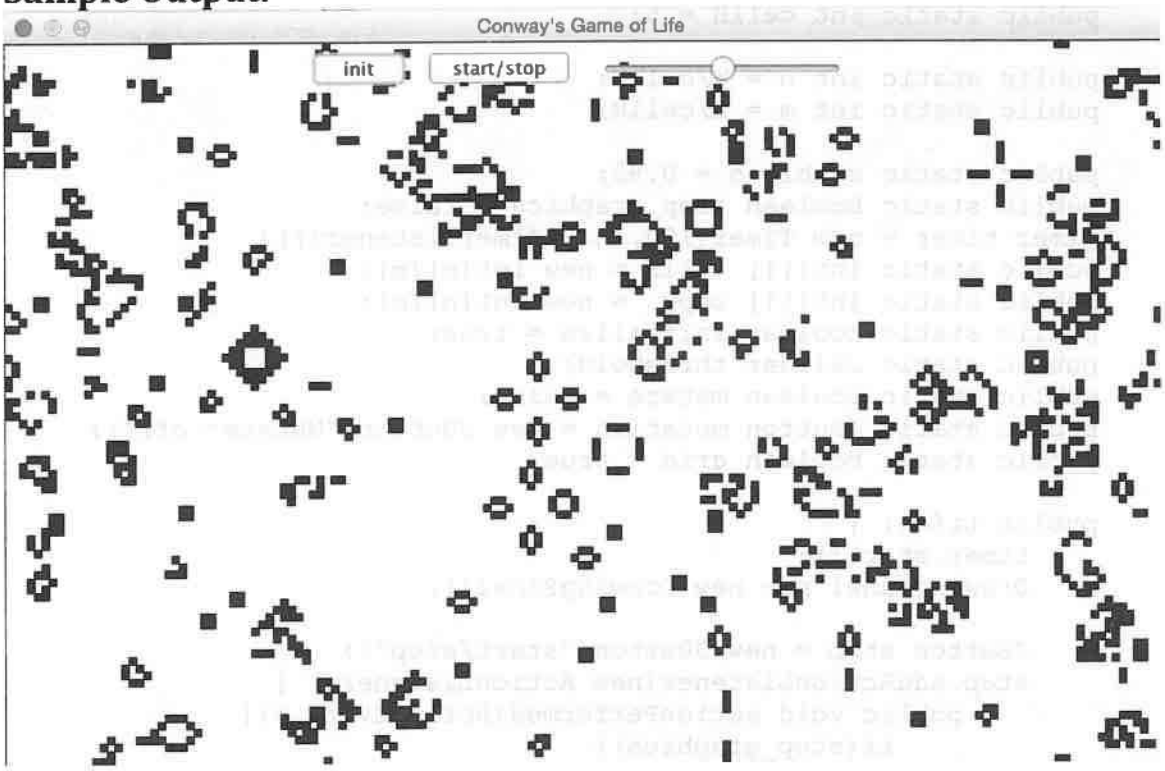
```
    }

    static class DrawingPanel extends JPanel {

        private Timer timer = new Timer(30, new TimerListener());
        int m = 20; //points per poly
        int n = 100; // number of polys
        double[][] x = new double[n][m];
        double[][] y = new double[n][m];
        double maxRadius = 40;
        double minRadius = 5;
        double[] xCenter = new double[n];
        double[] yCenter = new double[n];
        public Color[] color = new Color[n];

        public DrawingPanel() {
            initializePolygons();
            timer.start();
        }

        private void initializePolygons(){
            double theta;
            for (int i = 0; i < n; i++) {
                color[i] = randomColor();
                xCenter[i] = Math.random() * W;
                yCenter[i] = Math.random() * H;
                for (int k = 0; k < m; k++) {
                    theta = k * 2.0 * Math.PI / m;
                    double r = minRadius + (maxRadius - minRadius) *
Math.random();
                    //use this line if you want regular polygons:
                    //double r = maxRadius;
                    x[i][k] = r * Math.cos(theta) ;
                    y[i][k] = r * Math.sin(theta) ;
                }
            }
        }

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            updatePositions();
            drawPolygons(g);
        }

        private void updatePositions() {
            double alpha = 0.1;
            for(int i = 0; i < n; i++) {
                for(int k = 0; k < m; k++) {

                    double newX =  (x[i][k]) * Math.cos(alpha) +
(y[i][k]) * Math.sin(alpha);
                    double newY = -(x[i][k]) * Math.sin(alpha) +
(y[i][k]) * Math.cos(alpha);
```

```
                x[i][k] = newX ;
                y[i][k] = newY ;
            }
          //xCenter[i] += 1.0;
          //yCenter[i] += 1.0;
        }

    }

    class TimerListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            repaint();
        }
    }

    private void drawPolygons(Graphics g) {
        for (int i = 0; i < n; i++) {
            int[] polyX = new int[m];
            int[] polyY = new int[m];
            //Color color = randomColor();
            g.setColor(color[i]);
            for(int k = 0; k < m; k++) {
               polyX[k] = (int)(x[i][k] + xCenter[i]);
               polyY[k] = (int)(y[i][k] + yCenter[i]);
            }
            g.fillPolygon(polyX, polyY, m);
            g.setColor(Color.black);
            g.drawPolygon(polyX, polyY, m);
        }
    }

    private Color randomColor() {
        Color color = new Color(
                (int) (256 * Math.random()),
                (int) (256 * Math.random()),
                (int) (256 * Math.random())
        );
        return color;
    }
}

public static void main(String[] args) {
    RotatingPolygon frame = new RotatingPolygon();
    frame.setTitle("Drawings");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(W, H);
    frame.setVisible(true);
}
}
```

Java Project: Life

Description: This program simulates Conway's Game of Life. Using relatively simple rules, the cells change between black and white giving the impression of motion and other life-like activity. Various patterns emerge – very thought provoking. The program has a few settings that can be played with to get different effects, in particular a slide bar that allows for the density of white vs. black at the start of the simulation and "init" and "stop/start" buttons to restart or pause the simulation.

Sample Output:




```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package life;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.util.Random;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.Timer;

/**
 *
 * @author wwolfe3
 */
public class Life extends JFrame {

    public static int W = 800;
    public static int H = 500;

    public static int cellW = 5;
    public static int cellH = 5;

    public static int n = W/cellW;
    public static int m = H/cellH;

    public static double d = 0.92;
    public static boolean stop_graphics = false;
    Timer timer = new Timer(100, new TimerListener());
    public static int[][] cells = new int[n][m];
    public static int[][] copy  = new int[n][m];
    public static boolean initialize = true;
    public static JSlider threshold;
    public static boolean mutate = false;
    public static JButton mutation = new JButton("Mutate: off");
    public static boolean grid = true;

    public Life() {
        timer.start();
        DrawingPanel p = new DrawingPanel();

        JButton stop = new JButton("start/stop");
        stop.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                if(stop_graphics){
                    stop_graphics = false;
                } else {
                    stop_graphics = true;
                }
            }
        });

        JButton init = new JButton("init");
        init.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                initialize = true;
                stop_graphics = true;
            }
```

```
        });

        threshold = new JSlider(JSlider.HORIZONTAL, 0 , 100, 50);
        p.add(init);
        p.add(stop);
        p.add(threshold);
        this.add(p);

    }

    static class DrawingPanel extends JPanel {
        public DrawingPanel () {
            initializeCells();
        }

        public void initializeCells() {
            for(int j = 0; j < m; j++) {
                for(int i = 0; i < n; i++) {
                    if(Math.random() > d) {
                        cells[i][j] = 1;
                    } else {
                        cells[i][j] = 0;
                    }
                }
            }
        }

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if(initialize) {
                d = threshold.getValue()/100.0;
                initializeCells();
            }
            initialize = false;
            if(grid) {
                drawGrid(g);
            } else {

            }

            if(!stop_graphics) {
                createCopy();
                updateGrid();
            }
        }

        private void updateGrid() {
            for(int j = 0; j < m; j++) {
                for(int i = 0; i < n; i++) {
                    cells[i][j] = copy[i][j];
                }
            }
        }
```

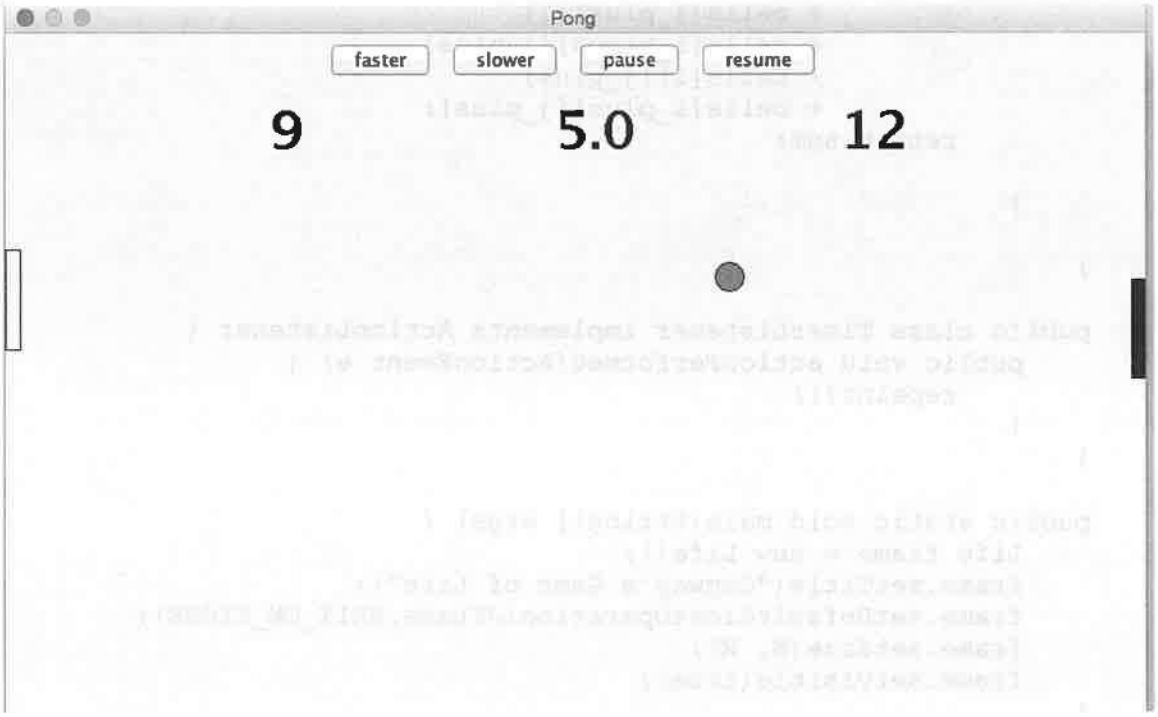
```
        private void createCopy() {
            for(int j = 0; j < m; j++) {
                for(int i = 0; i < n; i++) {
                  copy[i][j] = cells[i][j];
                  int count = getNeighborCount(i,j);
                  if (cells[i][j] == 1 && count < 2) {
                      copy[i][j] = 0;
                  }
                  if(cells[i][j] == 1 && (count == 2 || count == 3)
){
                      copy[i][j] = 1;
                  }

                  if(cells[i][j] == 1 && count > 3) {
                      copy[i][j] = 0;
                  }

                  if(cells[i][j] == 0 && count == 3) {
                      copy[i][j] = 1;
                  }
                }
            }
        }

        private void drawGrid(Graphics g) {
            Random rand = new Random();
            g.drawRect(0,0,W,H);
            for(int j = 0; j < m; j++) {
                for(int i = 0; i < n; i++) {
                    int x = i * cellW;
                    int y = j * cellH;
                    if(cells[i][j] == 1) {
                        //Color color = new Color(rand.nextInt(256),
rand.nextInt(256), rand.nextInt(256));
                        //g.setColor(color);
                        g.fillRect(x,y, cellW, cellH);
                        g.setColor(Color.BLACK);
                        g.drawRect(x,y, cellW, cellH);

                    }else {
                        //g.drawRect(x,y, cellW, cellH);
                    }
                    if (cells[i][j] == 0) {
                        if (Math.random() < 0.0001) cells[i][j] = 1;
                    }
                }
            }
        }

        public static int getNeighborCount(int i, int j) {
            int i_minus = (i - 1) % n;
            if(i_minus == -1) {
                i_minus = n - 1;
            }
```

```
            int i_plus = (i + 1) % n;

            int j_minus = (j - 1) % m;
            if(j_minus == -1) {
                j_minus = m-1;
            }

            int j_plus  = (j + 1) % m;

            int sum = cells[i_minus][j_minus]
                    + cells[i][j_minus]
                    + cells[i_plus][j_minus]
                    + cells[i_minus][j]
                    + cells[i_plus][j]
                    + cells[i_minus][j_plus]
                    + cells[i][j_plus]
                    + cells[i_plus][j_plus];
            return sum;

        }

    }

    public class TimerListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            repaint();
        }
    }

    public static void main(String[] args) {
        Life frame = new Life();
        frame.setTitle("Conway's Game of Life");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W, H);
        frame.setVisible(true);
    }

}
```

Java Project: Pong

Description: This programs simulates the game of Pong. It has controls to pause and speed up or slow down the speed of the ball. The P and L keys control the right paddle and the Q and A keys to control the left paddle.

Sample Output:




```
package pong;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import javax.swing.JButton;
import javax.swing.JFrame;
```

```
import javax.swing.JPanel;
import javax.swing.Timer;
import sun.audio.AudioPlayer;
import sun.audio.AudioStream;

public class Pong extends JFrame {

    public static int W = 800;
    public static int H = 500;

    public Pong() {
        JPanel canvas = new PongCanvas();
        canvas.requestFocus();
        add(canvas);
    }

    class PongCanvas extends JPanel {

        double ballRadius = 10;
        double ballX = W / 2;
        double ballY = H / 2;
        double theta = (Math.PI / 2.0) * Math.random() - Math.PI /
4.0;
        int toggle = 1;
        double ballSpeed = 5;
        double ballDx = ballSpeed * Math.cos(theta);
        double ballDy = ballSpeed * Math.sin(theta);
        double paddleW = 10;
        double paddleH = 70;
        double paddleLeftX = paddleW / 2;
        double paddleLeftY = H / 2;
        double paddleRightX = W - paddleW / 2;
        double paddleRightY = H / 2;
        double paddleLeftSpeed = 0;
        double paddleRightSpeed = 0;
        double paddleMaxSpeed = 4;
        private Timer timer = new Timer(10, new TimerListener());
        int playerLeftCount = 0;
        int playerRightCount = 0;
        char keyChar = 'A';
        private InputStream in;
        JButton faster = new JButton("faster");
        JButton slower = new JButton("slower");
        JButton pause = new JButton("pause");
        JButton resume = new JButton("resume");

        public PongCanvas() {
            addKeyListener(new KeyAdapter() {
                public void keyPressed(KeyEvent e) {
                    System.out.println("keyCode = " +
e.getKeyCode());
                    System.out.println("keyChar = " +
e.getKeyChar());
                    switch (e.getKeyChar()) {
```

```
                case 'q':
                    paddleLeftSpeed = -paddleMaxSpeed;
                    break;
                case 'a':
                    paddleLeftSpeed = +paddleMaxSpeed;
                    break;
                case 'p':
                    paddleRightSpeed = -paddleMaxSpeed;
                    break;
                case 'l':
                    paddleRightSpeed = +paddleMaxSpeed;
                    break;
                default:
                    keyChar = e.getKeyChar();
            }
        }

        public void keyReleased(KeyEvent e) {
            switch (e.getKeyChar()) {
                case 'q':
                    paddleLeftSpeed = 0;
                    break;
                case 'a':
                    paddleLeftSpeed = 0;
                    break;
                case 'p':
                    paddleRightSpeed = 0;
                    break;
                case 'l':
                    paddleRightSpeed = 0;
                    break;
                default:
                    keyChar = e.getKeyChar();
            }
        }
    });

    faster.addActionListener(new FasterListener());
    faster.setFocusable(false);
    this.add(faster);

    slower.addActionListener(new SlowerListener());
    slower.setFocusable(false);
    this.add(slower);

    pause.addActionListener(new PauseListener());
    pause.setFocusable(false);
    this.add(pause);

    resume.addActionListener(new ResumeListener());
    resume.setFocusable(false);
    this.add(resume);

    this.setFocusable(true);
```

```
        timer.start();

    }

    class FasterListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            ballSpeed++;
        }
    }

    class SlowerListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            ballSpeed--;
            if (ballSpeed <= 1) {
                ballSpeed = 1;
            }
        }
    }

    class PauseListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            timer.start();
        }
    }

    class ResumeListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            timer.stop();
        }
    }

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        theta = (Math.PI / 2.0) * Math.random() - Math.PI / 4.0;
        leftWallHit();
        rightWallHit();
        bottomHit();
        topHit();
        paddleLeftHit();
        paddleRightHit();
        updateBall();
        drawBall(g);
        updateLeftPaddle();
        drawLeftPaddle(g);
        updateRightPaddle();
        drawRightPaddle(g);
```

```
            Font font = new Font("New Courier", Font.BOLD, 36);
            g.setFont(font);

            FontMetrics metris = g.getFontMetrics(font);

            int hgt = metris.getHeight();

            String textLeft, textRight, textCenter;
            int adv;

            textLeft = "" + playerLeftCount;
            textRight = "" + playerRightCount;
            textCenter = "" + ballSpeed;
            adv = metris.stringWidth(textLeft);
            Dimension size = new Dimension(adv + 2, hgt + 2);
            g.drawString(textLeft, getWidth() / 4 - size.width / 2,
getHeight() / 8 + size.height / 2);
            g.drawString(textCenter, getWidth() / 2 - size.width /
2, getHeight() / 8 + size.height / 2);
            g.drawString(textRight, 3 * getWidth() / 4 - size.width
/ 2, getHeight() / 8 + size.height / 2);
        }

        private void updateLeftPaddle() {
            paddleLeftY += paddleLeftSpeed;
            if (paddleLeftY > getHeight() - paddleH / 2) {
                paddleLeftY = getHeight() - paddleH / 2;
            }
            if (paddleLeftY < paddleH / 2) {
                paddleLeftY = paddleH / 2;
            }
        }

        private void updateRightPaddle() {
            paddleRightY += paddleRightSpeed;
            if (paddleRightY > getHeight() - paddleH / 2) {
                paddleRightY = getHeight() - paddleH / 2;
            }

            if( paddleRightY < paddleH/2) {
                paddleRightY = paddleH/2;
            }
        }

        private void leftWallHit() {
            if (ballX - ballRadius <= 0) {
                playerRightCount++;
                playSound("miss2.wav");
                resetBall();
            }
        }

        private void rightWallHit() {
```

```
        if (ballX + ballRadius >= getWidth()) {
            playerLeftCount++;
            playSound("miss2.wav");
            resetBall();
        }
    }

    private void bottomHit() {
        if (ballY >= getHeight() - ballRadius) {
            ballDy = -ballDy;
            playSound("hit2.wav");
        }
    }

    private void topHit() {
        if (ballY - ballRadius < 0) {
            ballDy = -ballDy;
            playSound("hit2.wav");
        }
    }

    private void resetBall() {
        ballX = getWidth() / 2;
        ballY = getHeight() / 2;
        toggle = -1 * toggle;
        theta = (Math.PI / 2.0 * Math.random() - Math.PI / 4.0);
        if (toggle == -1) {
            theta += Math.PI;
        }
        ballDx = ballSpeed * Math.cos(theta);
        ballDy = ballSpeed * Math.sin(theta);
    }

    private void paddleLeftHit() {
        if ((ballX < paddleW + ballRadius)
                && (ballY < paddleLeftY + paddleH / 2)
                && (ballY > paddleLeftY - paddleH / 2)) {

            ballDx = -ballDx;
            ballDy += 2.0 * Math.random() - 1.0;
            playSound("hit2.wav");
        }
    }

    private void paddleRightHit() {
        if ((ballX > getWidth() - paddleW - ballRadius)
                && (ballY < paddleRightY + paddleH / 2)
                && (ballY > paddleRightY - paddleH / 2)) {
            ballDx = -ballDx;
            ballDy += 2.0* Math.random() - 1.0;
            playSound("hit2.wav");
        }
    }
```

```
        private void playSound(String fileName) {
            try {
                in = new FileInputStream(new File(fileName));
                AudioStream audioStream = new AudioStream(in);
                AudioPlayer.player.start(audioStream);

            } catch (Exception e) {
                System.out.println("problem withsound effect: " +
fileName);
            }
        }

        private void drawBall(Graphics g) {
            g.setColor(Color.red);
            g.fillOval((int) (ballX - ballRadius), (int) (ballY -
ballRadius), (int) (2 * ballRadius), (int) (2 * ballRadius));
            g.setColor(Color.black);
            g.drawOval((int) (ballX - ballRadius), (int) (ballY -
ballRadius), (int) (2 * ballRadius), (int) (2 * ballRadius));
        }

        private void drawLeftPaddle(Graphics g) {
            g.drawRect(0, (int) (paddleLeftY - paddleH / 2), (int)
paddleW, (int) (paddleH));
        }

        private void drawRightPaddle(Graphics g) {
            g.fillRect((int) (getWidth() - paddleW), (int)
(paddleRightY - paddleH / 2), (int) paddleW, (int) (paddleH));
        }

        private void updateBall() {
            ballX += ballDx;
            ballY += ballDy;
        }

        class TimerListener implements ActionListener {

            public void actionPerformed(ActionEvent e) {
                repaint();
            }
        }
    }

    public static void main(String[] args) {
        JFrame frame = new Pong();
        frame.setTitle("Pong");
        frame.setSize(W, H);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }

}
```

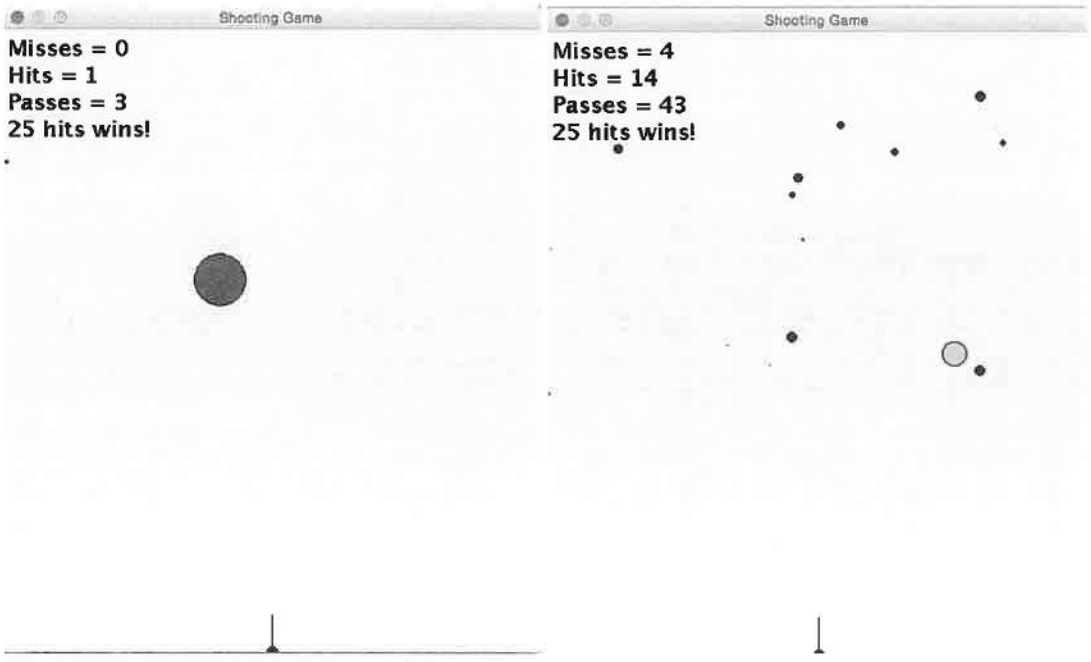
Java Project: ShootingGameNew

Description: This is a simple target shooting game. The player gets a shooter at the bottom of the screen. The player shoots with the up arrow key. Left and right arrows change the angle of the gun's turret. Targets are spawned, one at a time, from the top of the screen. If a bullet hits a target there is a simulated explosion and the player gains points.

Need a few files to make this work:

Ball.java: A simple class to define the targets in the game.
Particle.java: A class to model the bits and pieces that emerge when a target is hit and explodes.
ParticleSystem.java: A class to manage the many particles that are spawned at an explosion.
ShootingGameNew.java: The main class.

Sample Output:



```
package shootinggamenew;

import java.applet.AudioClip;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.Random;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.Timer;
import sun.audio.AudioPlayer;
import sun.audio.AudioStream;

/**
 * @author william wolfe -- idea from Liang's text, Chapter 16 A
ball falls from
 * above. you have an arrow-controlled gun at bottom of screen.
left-arrow:
 * swivel left right-arrow: swivel right up-arrow fires a shot in
direction gun
 * is pointing -- added a few variations: changing ball
size/color/speed --
 * "win" particle system -- "explosion" particle system
 */
public class ShootingGameNew extends JFrame {

    public static final int W = 500;
    public static final int H = 600;
    public static boolean shoot;

    public ShootingGameNew() {
        JPanel game = new GameCanvas();
        game.setFocusable(true);
        add(game);
    }

    class GameCanvas extends JPanel {

        public double gunW;
        public double gunH;
        public double theta;//gun pointing
        public double dTheta;//gun rotation rate
        public double speed;
        public boolean shot; // true when bullet in motion
```

```
        public Ball target;
        public Ball bullet;

        public double maxTargetSpeed = 20;
        public double maxTargetRadius = 30;

        public int misses = 0;
        public int hits = 0;
        public int passes = 0;
        public int bounce = 0;

        Timer timer = new Timer(10, new TimerListener());
        //sound:
        AudioClip clip;
        //particle system for explosions:
        ParticleSystem particleSystem;
        ParticleSystem particleSystemWin;

        public GameCanvas() {
            //initialize the gun parameters:
            gunW = 35;
            gunH = 5;
            theta = Math.PI / 2.0; //gun pointing straight up at
start
            dTheta = 0.1;//gun rotation rate
            speed = 20.0;//gun speed (or "power")
            shot = false;

            //initialize the traget parameters:
            target = new Ball();

            target.setRadius(25);
            resetTarget();

            //initialize the bullet parameters:
            bullet = new Ball();
            bullet.setRadius(5);
            resetBullet();

            //initialize the explosion particle system:
            particleSystem = new ParticleSystem(30, W / 2.0, H /
2.0);
            //initialize the "win" particle system:
            particleSystemWin = new ParticleSystem(500, W / 2, H /
2);
            particleSystemWin.name = "WIN";
            particleSystemWin.reset(W / 2, H / 2);
            particleSystemWin.maxR = 50;

            timer.start();

            this.addKeyListener(new KeyAdapter() {
                public void keyPressed(KeyEvent e) {
                    if (e.getKeyCode() == KeyEvent.VK_LEFT) {
```

```
                    theta += dTheta;
                    if (theta >= Math.PI) {
                        theta = Math.PI - 0.01;
                    }
                }
                if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
                    theta -= dTheta;

                    if (theta <= 0.0) {
                        theta = 0.01;
                    }
                }

                if (e.getKeyCode() == KeyEvent.VK_UP) {
                    if (!shot) {
                        InputStream in;
                        try {
                            in = new FileInputStream(new
File("audio/bullet.wav"));
                            AudioStream audioStream = new
AudioStream(in);

AudioPlayer.player.start(audioStream);
                        } catch (Exception soundException) {
                            JOptionPane.showMessageDialog(null,
soundException);
                        }

                        bullet.setVelocityPolar(theta, speed);
                        shot = true;
                    }

                }
            }
        });
    }

    public void updateGun() {
        //this does nothing right now -- for future mods
    }

    public void drawGun(Graphics g) {
        g.drawLine(
                getWidth() / 2,
                getHeight(),
                (int) (getWidth() / 2 + gunW * Math.cos(theta)),
                (int) (getHeight() - gunW * Math.sin(theta))
        );
        g.drawOval((int) (getWidth() / 2 - bullet.r), (int)
(getHeight() - bullet.r), (int) (2 * bullet.r), (int) (2 *
bullet.r));
    }

    public boolean collision() {
```

```
            if (target.distanceTo(bullet) < target.r + bullet.r) {
                InputStream in;
                try {
                    in = new FileInputStream(new
File("audio/hit2.wav"));
                    AudioStream audioStream = new AudioStream(in);
                    AudioPlayer.player.start(audioStream);
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null,
e.toString());
                }
                return true;
            } else {
                return false;
            }
        }

        public void drawScore(Graphics g) {
            Font font = new Font("New Courier", Font.BOLD, 20);
            g.setFont(font);
            g.setColor(Color.black);
            g.drawString("Misses = " + misses, 5, 25);
            g.drawString("Hits = " + hits, 5, 50);
            g.drawString("Passes = " + passes, 5, 75);
            g.drawString("25 hits wins!", 5, 100);
        }

        public void resetTarget() {
            //target.r = Math.random() * maxTargetRadius;
            if (getWidth() > 0) {
                target.x = getWidth() / 2.0;
            } else {
                target.x = W / 2.0;
            }

            target.y = target.r;
            target.dx = maxTargetSpeed * Math.random() -
maxTargetSpeed / 2;
            target.dy = maxTargetSpeed / 2 * Math.random();

            Random rand = new Random();
            target.color = new Color(rand.nextInt(256),
rand.nextInt(256), rand.nextInt(256));
        }

        public void resetBullet() {
            if (getWidth() > 0) {
                bullet.setPosition(getWidth() / 2.0, getHeight());
            } else {
                bullet.setPosition(W / 2.0, H);
            }

            bullet.setVelocityPolar(theta, 0);
            shot = false;
```

```
}

public int checkBounce(Ball ball) {
    //check target against East wall:
    if (ball.x >= getWidth() - ball.r) {
        return -1;
    }
    //check target against West wall:
    if (ball.x <= ball.r) {
        return +1;
    }
    return 0;
}

public void checkPass(Ball target) {
    //Check target against South wall:
    if (target.y >= getHeight() + target.r) {
        passes++;
        resetTarget();
    }
}

public void checkMiss(Ball bullet) {
    //Check bullet against the North wall
    if (bullet.y - bullet.r < 0) {
        misses++;
        resetBullet();
    }
}

protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    //check for target bounce off the side walls:
    bounce = checkBounce(target);
    if (bounce == -1) {
        target.dx = -Math.abs(target.dx);
    } else if (bounce == +1) {
        target.dx = +Math.abs(target.dx);
    }
    checkPass(target);
    target.move();

    //check for bullet bounce off side walls:
    bounce = checkBounce(bullet);
    if (bounce == -1) {
        bullet.dx = -Math.abs(bullet.dx);
    } else if (bounce == +1) {
        bullet.dx = +Math.abs(bullet.dx);
    }
    checkMiss(bullet);
    bullet.move();

    updateGun();
```

```
            particleSystem.update();

            if (collision()) {
                //initiate explosion at target.x, target.y:
                particleSystem.reset(target.x, target.y);
                particleSystem.setColor(target.color);
                //reset Target and Bullet:
                resetTarget();
                resetBullet();
                //bullet hit the target:
                hits++;
                //make the target harder it hit:
                target.r -= 1.0;
            }

            target.draw(g);
            bullet.draw(g);
            drawGun(g);
            particleSystem.draw(g);
            drawScore(g);
            //check for win:
            if (hits == 25) {
                particleSystemWin.setColor(Color.black);
                particleSystemWin.update();
                particleSystemWin.draw(g);
                setBackground(Color.red);
                target.r = 25;
            }
        }

        class TimerListener implements ActionListener {

            public void actionPerformed(ActionEvent e) {
                repaint();
            }
        }
    }

    public static void main(String[] args) {
        JFrame frame = new ShootingGameNew();
        frame.setTitle("Shooting Game");
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(W, H);
    }
}
```

```
////////////////////////////////////////////////
package shootinggamenew;

import java.awt.Color;
import java.awt.Graphics;

public class Ball {
```

```
    public double x;
    public double y;
    public double r;
    public double dx;
    public double dy;
    public Color color;

    public Ball() {
        this.x = 0;
        this.y = 0;
        this.dx = 0;
        this.dy = 0;
        this.r = 25;
        this.color = Color.black;
    }

    public void draw(Graphics g) {
        g.setColor(color);
        g.fillOval((int) (x - r), (int) (y - r), (int) (2 * r),
(int) (2 * r));
        g.setColor(Color.black);
        g.drawOval((int) (x - r), (int) (y - r), (int) (2 * r),
(int) (2 * r));
    }

    public void move() {
        x += dx;
        y += dy;
    }

    public void move(double dx, double dy) {
        x += dx;
        x += dy;
    }

    public void moveTo(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public void setPosition(double x, double y) {
        this.x = x;
        this.y = y;

    }

    public void setColor(Color color) {
        this.color = color;
    }

    public void setRadius(double r) {
        this.r = r;
    }
```

```
    public double distanceTo(Ball b) {
        double distance = Math.sqrt(Math.pow(this.x - b.x, 2.0) +
Math.pow(this.y - b.y, 2.0));
        return distance;
    }

    public void setVelocityCartesian(double dx, double dy) {
        this.dx = dx;
        this.dy = dy;
    }

    public void setVelocityPolar(double theta, double speed) {
        this.dx = speed * Math.cos(theta);
        this.dy = -speed * Math.sin(theta);
    }
}
```

```
////////////////////////////////////////////////////
package shootinggamenew;

import java.awt.Color;
import java.awt.Graphics;

public class Particle {

    public double x;
    public double y;
    public double dx;
    public double dy;
    public double speed;
    public Color color;
    public double r;
    public boolean isActive;
    public String name;

    public Particle() {
        x = 0;
        y = 0;
        dx = 0;
        dy = 0;
        speed = 0;
        color = new Color((int)(256 * Math.random()),
(int)(256*Math.random()), (int)(256*Math.random()));
        r = 1;
        isActive = false;
        name = "";
    }

    public void draw(Graphics g) {
        g.setColor(color);
        g.fillOval((int) (x - r), (int) (y - r), (int) (2 * r),
(int) (2 * r));
        g.setColor(Color.BLACK);
```

```
        g.drawOval((int) (x - r), (int) (y - r), (int) (2 * r),
(int) (2 * r));
        g.drawString(name, (int)x,(int)y);

    }

    public void reset() {
        isActive = false;
    }

    public void update() {
        x += dx;
        y += dy;
    }

    public void setColor(Color color){
        this.color = color;
    }

    public String toString() {
        String out = "";
        out = "[" + x + ", " + y + ", " + dx + ", " + dy + ", " +
color.toString() + "]";
        return out;
    }

}
```

```
////////////////////////////////////////////////
package shootinggamenew;

import java.awt.Color;
import java.awt.Graphics;

public class ParticleSystem {

    Particle[] particles;
    public double maxR = 5;
    public double maxSpeed = 5;
    public int n;
    public String name = "";

    ParticleSystem(int n, double x, double y) {
        this.n = n;
        particles = new Particle[n];
        for (int i = 0; i < n; i++) {
            particles[i] = new Particle();
            particles[i].isActive = false;
            particles[i].x = x;
            particles[i].y = y;
            particles[i].dx = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].dy = 2 * maxSpeed * Math.random() -
maxSpeed;
```

```
            particles[i].r = maxR * Math.random();
            particles[i].name = this.name;
        }
    }

    public void setColor(Color c) {
        for (int i = 0; i < n; i++) {
            particles[i].color = c;
        }
    }

    public void draw(Graphics g) {
        for (int i = 0; i < n; i++) {
            if (particles[i].isActive) {
                particles[i].draw(g);
            }

        }
    }

    public void reset(double x, double y) {

        for (int i = 0; i < n; i++) {
            particles[i] = new Particle();
            particles[i].isActive = true;
            particles[i].x = x;
            particles[i].y = y;
            particles[i].dx = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].dy = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].r = maxR * Math.random();
            particles[i].name = this.name;
        }
    }

    public void update() {
        for (Particle p : particles) {
            p.update();
        }
    }

    public String toString() {
        String out = "";
        for (int i = 0; i < n; i++) {
            out += " " + particles[i].toString();
        }
        return out;
    }
}
```

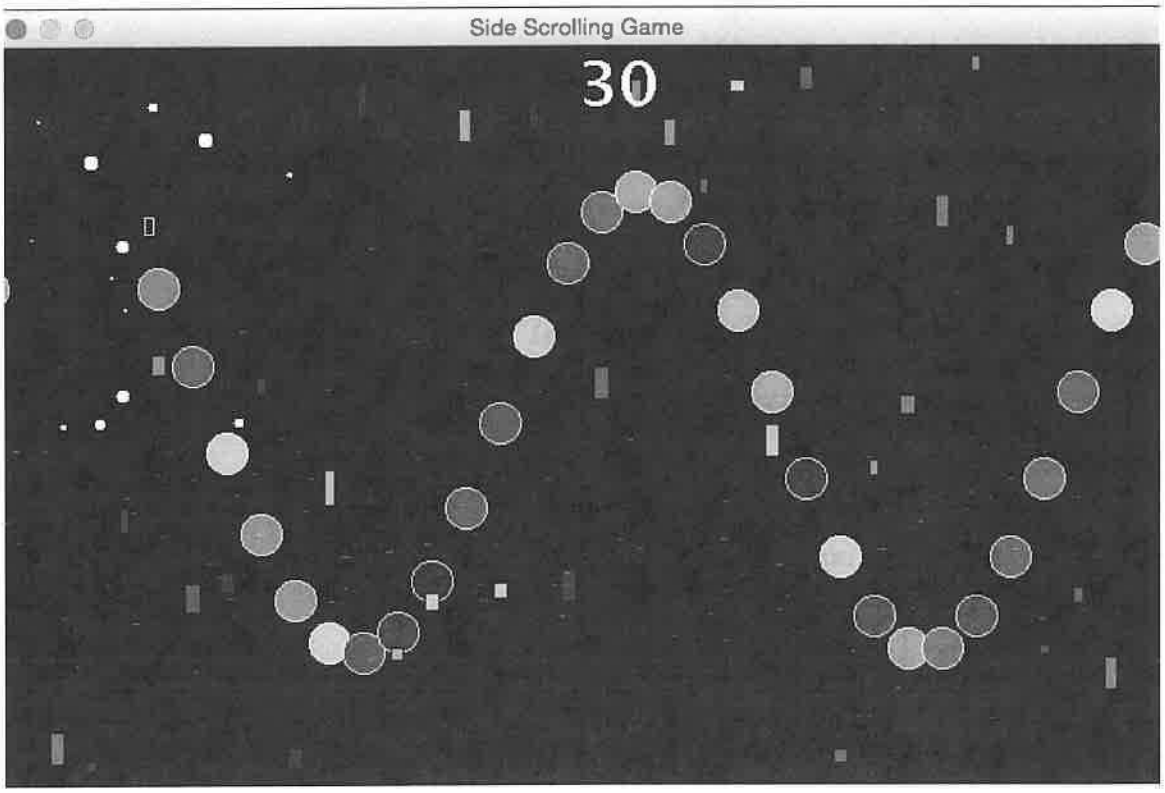
Java Project: SideScrollGame

Description: This sidescroll game challenges the player to collect certain objects while avoiding the obstacles. The added feature of the game is that it automatically gets harder as the player's score increases and gets correspondingly easier as the player's score goes down.

The collectables are colored balls, and the obstacles are rectangles. They change in size as the player earns or loses points.

The player is a small white, hollow, rectangle, controlled by the arrow keys.

Sample Output:



```
package sidescrollgame;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;
import sun.audio.AudioData;
import sun.audio.AudioPlayer;
import sun.audio.AudioStream;
import sun.audio.ContinuousAudioDataStream;

public class SideScrollGame extends JFrame {

    public final int W = 400;
    public final int H = 400;
    public int nEvents = 0;

    public SideScrollGame() {
        JPanel canvas = new Canvas();
        add(canvas);
    }

    public class Canvas extends JPanel {

        public ArrayList<Ball> balls = new ArrayList<Ball>();
        public ArrayList<Obstacle> obstacles = new
ArrayList<Obstacle>();
        public Player player;
        public int timerRate = 10;
        public int spawnSpacing = 20;
        public double maxBallRadius = 12.0;
        public double minBallRadius = 12.0;
        public int minObstacleW = 5;
        public int maxObstacleW = 10;
        public int minObstacleH = 5;
        public int maxObstacleH = 20;
        public double speed = 1.0;

        public double f = 0.3;
        private InputStream in;
        public int score = 0;
        int ballHitPoints = 10;
```

```
int obstacleHitPoints = 20;

//for the esplosion, we need "debris":
public int nDebris = 20;
public Color debrisColor = Color.WHITE;
public double[] debrisX = new double[nDebris];
public double[] debrisY = new double[nDebris];
public double[] debrisDX = new double[nDebris];
public double[] debrisDY = new double[nDebris];
public double[] debrisR = new double[nDebris];
public boolean[] isActive = new boolean[nDebris];

public Canvas() {
    initializeBalls();
    initializePlayer();
    resetDebris();
    Timer timer = new Timer(timerRate, new TimerListener());
    timer.start();
    //playes an audo file:
    playSound("audio/DeadlySkirmish(loop).wav");

    addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            switch (e.getKeyCode()) {
                case 37://left arrow
                    player.dx = -player.speed;
                    break;
                case 39://right arrow
                    player.dx = +player.speed;
                    break;
                case 38://up arrow
                    player.dy = -player.speed;
                    break;
                case 40: //down arrow
                    player.dy = +player.speed;
                    break;
                default:
                // keyChar = e.getKeyChar();
            }
        }

        public void keyReleased(KeyEvent e) {
            switch (e.getKeyCode()) {
                case 37:
                    player.dx = 0;
                    player.dy = 0;
                    break;
                case 39:
                    player.dx = 0;
                    player.dy = 0;
                    break;
                case 38:
                    player.dy = 0;
                    player.dx = 0;
```

```
                        break;
                    case 40:
                        player.dy = 0;
                        player.dx = 0;
                        break;
                    default:
                    //keyChar = e.getKeyChar();
                }
            }
        });
        setFocusable(true);
    }

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        setBackground(Color.black);
        if (nEvents % spawnSpacing == 0) {
            spawnBall();
            spawnObstacle();
        }
        setMaxObstacleHeight();
        updatePlayer();
        updateBalls();
        updateObstacles();
        updateDebris();
        if (collision()) {
            explosion();
        }
        drawPlayer(g);
        drawBalls(g);
        drawObstacles(g);
        drawDebris(g);
        drawScore(g);
    }

    public void setMaxObstacleHeight() {
        //an ad-hoc way to make the game harder as the player
get more points:
        if (score < 100) {
            maxObstacleH = getHeight() / 20;
            setBallRadius(maxBallRadius);
        }
        if (score < 0) {
            maxObstacleH = minObstacleH + 1;
            setBallRadius(2 * maxBallRadius);
        }

        if (score < -50) {
            maxObstacleH = minObstacleH + 1;
            setBallRadius(4 * maxBallRadius);
        }
        if (score > 100) {
            maxObstacleH = getHeight() / 10;
            setBallRadius(maxBallRadius / 2);
```

```
            }
            if (score == 100) {
                //playSound("audio/Coin_Pick_Up_03.wav");
                //playSound("audio/bullet.wav");
            }
            if (score > 200) {
                maxObstacleH = getHeight() / 5;
                setBallRadius(maxBallRadius / 3);
            }
            if (score == 200) {
                //playSound("audio/Coin_Pick_Up_03.wav");
                //playSound("audio/miss2.wav");
            }

            if (score > 300) {
                maxObstacleH = getHeight() / 2;
                setBallRadius(maxBallRadius / 4);
            }
            if (score == 300) {
                //playSound("audio/Coin_Pick_Up_03.wav");
                //playSound("audio/hit2.wav");
            }
            if (score > 400) {
                maxObstacleH = getHeight() / 2;
                setBallRadius(maxBallRadius / 6);
            }
            if (score == 400) {
                //playSound("audio/Coin_Pick_Up_03.wav");
               playSound("audio/hit2.wav");
            }
        }

        public void drawScore(Graphics g) {
            Font f = new Font("New Courier", Font.BOLD, 36);
            g.setColor(Color.white);
            g.setFont(f);
            g.drawString("" + score, getWidth() / 2, f.getSize());
        }

        public boolean collision() {
            boolean return_value = false;
            for (int i = balls.size() - 1; i > -1; i--) {
                double distance = Math.sqrt((player.x -
balls.get(i).x) * (player.x - balls.get(i).x)
                        + (player.y - balls.get(i).y) * (player.y -
balls.get(i).y));
                if (distance <= player.h / 2 + balls.get(i).r) {
                    return_value = true;
                    balls.remove(i);
                    playSound("audio/hit2.wav");
                    score += ballHitPoints;
                }
            }
```

```
            for (int i = obstacles.size() - 1; i > -1; i--) {

                boolean xAxis = intervalOverlap(
                        player.x - player.w / 2,
                        player.x + player.w / 2,
                        obstacles.get(i).x - obstacles.get(i).w / 2,
                        obstacles.get(i).x + obstacles.get(i).w / 2
                );

                boolean yAxis = intervalOverlap(
                        player.y - player.h / 2,
                        player.y + player.h / 2,
                        obstacles.get(i).y - obstacles.get(i).h / 2,
                        obstacles.get(i).y + obstacles.get(i).h / 2
                );

                if (xAxis && yAxis) {
                    return_value = true;
                    obstacles.remove(i);
                    playSound("audio/miss2.wav");
                    score -= obstacleHitPoints;
                }

            }
            return return_value;
        }

        public boolean intervalOverlap(double a, double b, double c,
double d) {
            if ((c > a && c < b) || (d > a && d < b)) {
                return true;
            } else if (c < a && d > b) {
                return true;

            } else {
                return false;
            }
        }

        private void playSound(String fileName) {
            try {
                in = new FileInputStream(new File(fileName));
                AudioStream audioStream = new AudioStream(in);
                AudioPlayer.player.start(audioStream);

            } catch (Exception e) {
                System.out.println("problem with sound effect: " +
fileName);
            }
        }

        private void playBackgroundSound(String fileName) {
            sun.audio.ContinuousAudioDataStream loop = null;
            AudioStream audioStream;
```

```
            AudioData AD;
            try {//note: this dooes not work with a larger than 1MB
file:
                //It will not make a data object from audio streams
                //using a file larger than 1 MB.
                //input stream:
                in = new FileInputStream(new File(fileName));
                audioStream = new AudioStream(in);
                AD = audioStream.getData();
                loop = new ContinuousAudioDataStream(AD);

            } catch (Exception e) {
                System.out.println("problem with sound effect: " +
fileName);
                System.out.println("error: " + e.toString());
            }
            AudioPlayer.player.start(loop);
        }

        public void drawPlayer(Graphics g) {
            g.setColor(player.color);
            g.drawRect((int) (player.x - player.w / 2), (int)
(player.y - player.h / 2), (int) player.w, (int) player.h);
        }

        public void updatePlayer() {

            player.x += player.dx;
            player.y += player.dy;
            if (player.x < player.w / 2) {
                player.x = player.w / 2;
            }

            if (player.x > getWidth() - player.w / 2) {
                player.x = getWidth() - player.w / 2;
            }

            if (player.y < player.h / 2) {
                player.y = player.h / 2;
            }

            if (player.y > getHeight() - player.h / 2) {
                player.y = getHeight() - player.h / 2;
            }
        }

        public void initializePlayer() {
            player = new Player();
            player.w = 5;
            player.h = 10;
            player.x = player.w / 2.0;
            player.y = H / 2.0;
            player.dx = 0;
            player.dy = 0;
```

```
            player.speed = 1;
            player.color = Color.white;
        }

        public void spawnBall() {
            Ball b = new Ball();
            double t = nEvents * timerRate / 1000.0;
            double A = getHeight() / 3.0 - maxBallRadius;
            b.r = minBallRadius + (maxBallRadius - minBallRadius) *
Math.random();
            b.x = getWidth();
            b.y = getHeight() / 2.0 - A * Math.sin(2 * Math.PI * f *
t);
            b.speed = speed;
            b.dx = -b.speed;
            b.dy = 0.0;
            b.color = getRandomColor();
            balls.add(b);
        }

        public void spawnObstacle() {
            Obstacle obstacle = new Obstacle();
            obstacle.x = getWidth();
            obstacle.y = getRandom(0, getHeight());
            obstacle.color = getRandomColor();
            obstacle.speed = speed;
            obstacle.dx = -obstacle.speed;
            obstacle.dy = 0;
            obstacle.w = (double) getRandom(minObstacleW,
maxObstacleW);
            obstacle.h = (double) getRandom(minObstacleH,
maxObstacleH);
            obstacles.add(obstacle);
        }

        public Color getRandomColor() {
            Random r = new Random();
            return new Color(r.nextInt(256), r.nextInt(256),
r.nextInt(256));
        }

        public void drawBalls(Graphics g) {
            for (Ball b : balls) {
                g.setColor(b.color);
                g.fillOval((int) (b.x - b.r), (int) (b.y - b.r),
(int) (2 * b.r), (int) (2 * b.r));
                g.setColor(Color.white);
                g.drawOval((int) (b.x - b.r), (int) (b.y - b.r),
(int) (2 * b.r), (int) (2 * b.r));
            }
        }

        public void setBallRadius(double r) {
            for (Ball b : balls) {
```

```
            b.r = r;
        }
    }

    public void drawObstacles(Graphics g) {
        for (Obstacle obstacle : obstacles) {
            g.setColor(obstacle.color);
            g.fillRect(
                    (int) (obstacle.x - obstacle.w / 2),
                    (int) (obstacle.y - obstacle.h / 2),
                    (int) obstacle.w,
                    (int) obstacle.h
            );
            g.setColor(Color.black);
            g.drawRect(
                    (int) (obstacle.x - obstacle.w / 2),
                    (int) (obstacle.y - obstacle.h / 2),
                    (int) obstacle.w,
                    (int) obstacle.h
            );
        }
    }

    public void updateBalls() {
        for (int i = balls.size() - 1; i > -1; i--) {
            balls.get(i).x += balls.get(i).dx;
            if (balls.get(i).x <= -balls.get(i).r) {
                balls.remove(i);
            }
        }
    }

    public void updateObstacles() {
        for (int i = obstacles.size() - 1; i > -1; i--) {
            obstacles.get(i).x += obstacles.get(i).dx;
            if (obstacles.get(i).x <= -obstacles.get(i).w) {
                obstacles.remove(i);
            }
        }
    }

    public void initializeBalls() {
        Ball b = new Ball();
        b.r = maxBallRadius;
        b.x = W;
        b.y = H / 2.0;
        b.speed = 5.0;
        b.dx = -b.speed;
        b.dy = 0.0;
        balls.add(b);
    }

    public void initializeObstacles() {
        Obstacle obstacle = new Obstacle();
```

```
            obstacle.x = getWidth();
            obstacle.y = getRandom(0, getHeight());
            obstacle.speed = 5.0;
            obstacle.dx = -obstacle.speed;
            obstacle.dy = 0.0;
            obstacle.w = (double) getRandom(minObstacleW,
maxObstacleW);
            obstacle.h = (double) getRandom(minObstacleH,
maxObstacleH);
            obstacle.color = Color.black;
            obstacles.add(obstacle);
        }

        public void explosion() {
            for (int i = 0; i < nDebris; i++) {
                isActive[i] = true;
            }
            for (int i = 0; i < nDebris; i++) {
                debrisX[i] = player.x;
                debrisY[i] = player.y;
                debrisDX[i] = 10 * Math.random() - 5;
                debrisDY[i] = 10 * Math.random() - 5;
                debrisR[i] = 5 * Math.random();
            }
        }

        public void resetDebris() {
            for (int i = 0; i < nDebris; i++) {
                isActive[i] = false;
            }
        }

        public void updateDebris() {
            for (int i = 0; i < nDebris; i++) {
                if (debrisX[i] - debrisR[i] <= 0
                        || debrisX[i] + debrisR[i] >= W) {
                    isActive[i] = false;
                }
                if (debrisY[i] + debrisR[i] >= H
                        || debrisY[i] - debrisR[i] <= 0) {
                    isActive[i] = false;
                }
                debrisX[i] += debrisDX[i];
                debrisY[i] += debrisDY[i];
            }
        }

        public void drawDebris(Graphics g) {
            for (int i = 0; i < nDebris; i++) {
                if (isActive[i]) {
                    g.setColor(debrisColor);
                    g.fillOval((int) (debrisX[i] - debrisR[i]),
                            (int) (debrisY[i] - debrisR[i]),
                            (int) (2 * debrisR[i]),
```

```
                            (int) (2 * debrisR[i]));
                    }
                }
            }

    }

    public int getRandom(int p, int q) {
        Random r = new Random();
        return p + r.nextInt(q - p);
    }

    public class TimerListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            nEvents++;
            repaint();
        }
    }

    public static void main(String[] args) {
        JFrame frame = new SideScrollGame();
        frame.setTitle("Side Scrolling Game");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //frame.setLocationRelativeTo(null);
        frame.setSize(400, 400);
        frame.setVisible(true);
    }
}

//////////////////////////////////////////////////////////

package sidescrollgame;

import java.awt.Color;

public class Ball {
    public double x;
    public double y;
    public double dx;
    public double dy;
    public double speed;
    public double r;
    public Color color;
    public boolean visible;

    public Ball(){
        x = 0;
        y = 0;
        dx = 0;
        dy = 0;
        speed = 0;
        r = 1.0;
```

```
        color = Color.red;
        visible = true;
    }
}
//////////////////////////////////////////////
package sidescrollgame;

import java.awt.Color;

public class Obstacle {

    public double x;
    public double y;
    public double dx;
    public double dy;
    public double speed;
    public Color color;
    public boolean visible;
    public double w;
    public double h;

    public Obstacle() {
        x = 0;
        y = 0;
        dx = 0;
        dy = 0;
        speed = 0;
        color = Color.red;
        visible = true;
        w = 1;
        h = 1;
    }
}
//////////////////////////////////////////////

package shootinggamenew;

import java.awt.Color;
import java.awt.Graphics;

public class ParticleSystem {

    Particle[] particles;
    public double maxR = 5;
    public double maxSpeed = 5;
    public int n;
    public String name = "";

    ParticleSystem(int n, double x, double y) {
        this.n = n;
        particles = new Particle[n];
        for (int i = 0; i < n; i++) {
            particles[i] = new Particle();
            particles[i].isActive = false;
```

```
            particles[i].x = x;
            particles[i].y = y;
            particles[i].dx = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].dy = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].r = maxR * Math.random();
            particles[i].name = this.name;
        }
    }

    public void setColor(Color c) {
        for (int i = 0; i < n; i++) {
            particles[i].color = c;
        }
    }

    public void draw(Graphics g) {
        for (int i = 0; i < n; i++) {
            if (particles[i].isActive) {
                particles[i].draw(g);
            }

        }
    }

    public void reset(double x, double y) {

        for (int i = 0; i < n; i++) {
            particles[i] = new Particle();
            particles[i].isActive = true;
            particles[i].x = x;
            particles[i].y = y;
            particles[i].dx = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].dy = 2 * maxSpeed * Math.random() -
maxSpeed;
            particles[i].r = maxR * Math.random();
            particles[i].name = this.name;
        }
    }

    public void update() {
        for (Particle p : particles) {
            p.update();
        }
    }

    public String toString() {
        String out = "";
        for (int i = 0; i < n; i++) {
            out += " " + particles[i].toString();
        }
        return out;
```

Java Project: MadLib5

Description: This program mimics the classical "Madl-Lib" game. But instead of having the player choose words from certain categories, the computer uses lists of possible words and substitutes them in specified places in the original literature.

The setup begins by selecting a piece of literature (original text) and then tagging certain words for replacement. The tags are associated with files containing possible substitutions.

After the original file is "tagged", the tagged version is referred to as the "Template". The computer program runs through the template file looking for tags. When a tag is found, a random word is chosen from the the corresponding options file, and then substituted for the tagged word. The final output is referred to as the "mad-lib" version of the original text.

The "tags.txt" file lists all the possible tags that might be found in the template file. There is a file for each tag that lists all the words than can be substituted for that tag.

For example, the tags.txt file might contain such words as:

aching
adjective
adverb
amount
animal
another
bands

And, the template file might have something like:

The <aching> feeling was <adjective> to the <bands> of the <animal> etc.

The program will detect the tagged word <aching> and then look for a file with that name (aching.txt), which might look like this:

aching
shaking
throbbing
singing
yelling
hearing
seeing
being
having
etc.

Each of the listed words in the aching.txt file can be substituted for the tagged word.

The same process is followed for each of the tags found in the template file.

As you can see, the words in aching.txt are not necessarily synonyms, and in fact might be antonyms, or any other word that might substitute for the original tagged word in any way the user wants to allow or imagine.

This program therefore relies on several additional text files that list the tags and tag substitutions.

For a given piece of literature (original text) , the user must decide where to place tags to create the template file. And, the user must define a text file for each tag.

This project was easy when I first starting using grammatical tags such as <noun> and <adjective> and became more challenging, and more fun, when I extended it to tags that capture semantic variations.

Here are two sample outputs (A Christmas Carol and The Raven):
Original piece of Literature:

Mad Lib

MadLibs Originals

A Christmas Carol

Marley was dead: to begin with. There is no doubt whatever about that. The register of his burial was signed by the clergyman, the clerk, the undertaker, and the chief mourner. Scrooge signed it: and Scrooge's name was good upon 'Change, for anything he chose to put his hand to. Old Marley was as dead as a door-nail. Mind! I donÕt mean to say that I know, of my own knowledge, what there is particularly dead about a door-nail. I might have been inclined, myself, to regard a coffin-nail as the deadest piece of ironmongery in the trade. But the wisdom of our ancestors is in the simile; and my unhallowed hands shall not disturb it, or the CountryÕs done for. You will therefore permit me to repeat, emphatically, that Marley was as dead as a door-nail.

Madlib Version:

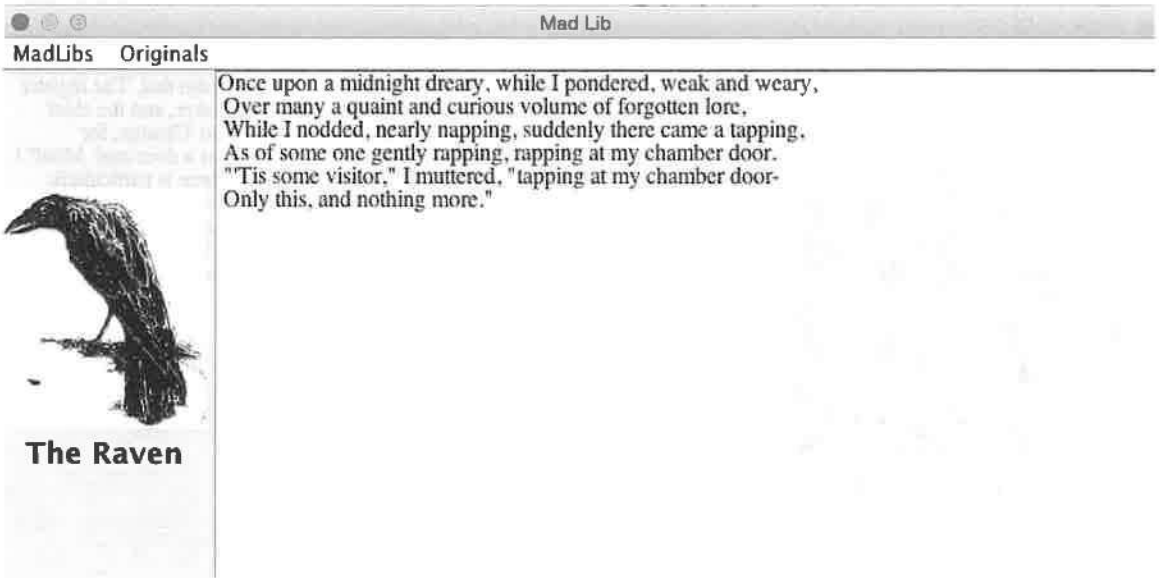
Mad Lib

MadLibs Originals

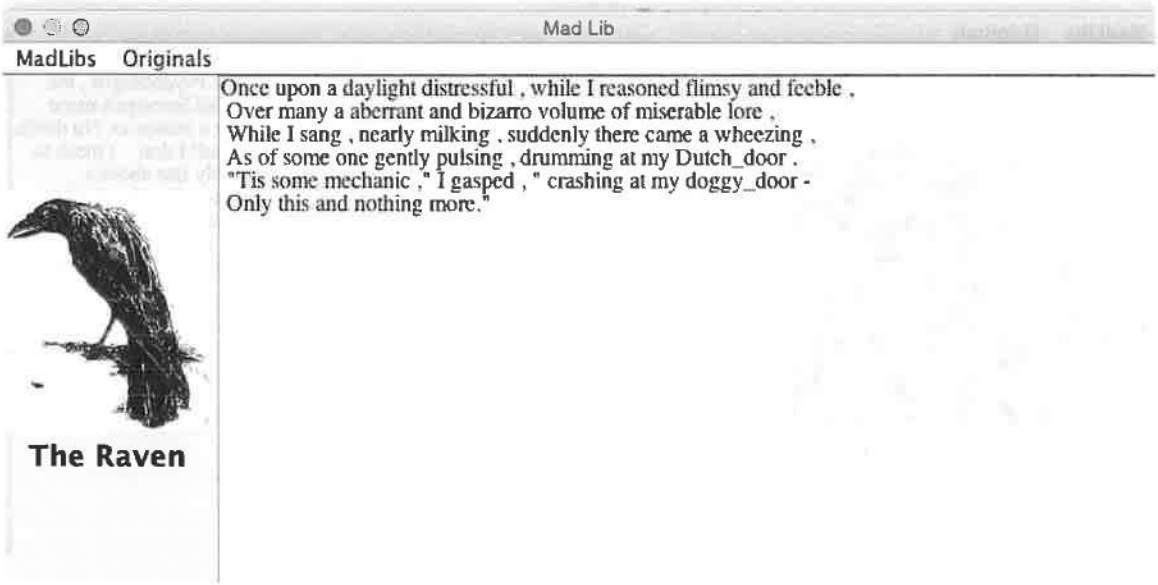
A Christmas Carol

Dewayne was sad : to begin with. There is no Sadness whatever about that. The chopping_block of the hanger was signed by the Surveyor , the Psychologist , the Teacher , and the chief Accountant . Even Scrooge signed it: and Scrooge's name was atrocious upon DVD_player , for anything he chose to put a senses to. No doubt about it, old Malena was as thoughtful as a bar_accessory . Mind! I don t mean to say that I know, of my own knowledge, what there is particularly late about a door-nail. I might have been inclined, myself, to regard a coffin-nail as the deadest piece of ironmongery in the trade. But the wisdom of our ancestors is in the simile; and my unhallowed hands shall not disturb it, or the Country s done for. You will therefore permit me to repeat, emphatically, that Marley was as dead as a door-nail.

Original text:



Madlib version:




```
package madlib5;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class MadLib5 extends JFrame {

    private StoryPanel storyPanel = new StoryPanel();
    private String[] fullName = new String[]{"Friendship", "I
Loved", "Declaration of Independence","Not in Vain", "The Raven", "A
Christmas Carol", "The Tiger", "The Great Gatsby", "Where the
Sidewalk Ends", "The Road Not Taken", "How a Toilet Works", "Tale of
Two Cities"};
    private String[] shortName = new String[] {"Friendship","Loved",
"Declaration", "Vain", "Raven", "Christmas", "Tiger", "Gatsby",
"Sidewalk", "Road", "Toilet", "Tale"};
    private String[] author = new String[]{"Lucius  Seneca",
"Langston Hughes", "Ben Franklin", "Emily Dickinson", "Edgar Allan
Poe", "Charles Dickens", "William Blake", "F. Scott Fitzgerald",
"Shel Silverstein", "Robert Frost", "How Stuf Works", "Charles
Dickens"};
    public char[] madlibsShortcuts = {'F','L', 'N', 'V', 'R', 'O',
'T', 'G', 'S', 'D', 'Q', 'H', 'C'};
    public char[] originalsShortcuts = {'F', 'L','N', 'V', 'R', 'O',
'T', 'G', 'S', 'D', 'Q', 'H', 'C'};
    public String storyTemplateFile = "";
    public String storyTitle = "";
    public String story = "";
    public static ArrayList<String> tags = new ArrayList<String>();

    public static void main(String[] args) throws
FileNotFoundException {
        MadLib5 frame = new MadLib5();
        frame.pack();
        frame.setTitle("Mad Lib");
        frame.setSize(800, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

    public MadLib5() throws FileNotFoundException {
        //read all the tags into an array list:
        Scanner fileScanner = new Scanner(new File("tags.txt"));
        while (fileScanner.hasNext()) {
            tags.add(fileScanner.next());
        }
        fileScanner.close();

        //initialize the display so the first story pops up at
start:
        storyTitle = fullName[0];
        storyTemplateFile = shortName[0] + "Original.txt";
        String story = getStory(storyTemplateFile);
```

```
        storyPanel.setImageIcon(new ImageIcon("images/" +
shortName[0] + ".jpg"));
        storyPanel.setTitle(storyTitle);
        storyPanel.setStory(story);
        setLayout(new BorderLayout());
        add(storyPanel, BorderLayout.CENTER);

        //Now, build the menu bar and selections:
        JMenuBar myMenuBar = new JMenuBar();
        //use this listener for all the menu selections:
        MenuListener myListener = new MenuListener();
        //first menu, for the madlibs:
        JMenu madlibs = new JMenu("MadLibs");
        for (int i = 0; i < fullName.length; i++) {
            JMenuItem item = new JMenuItem(fullName[i] + "
(MadLib)", madlibsShortcuts[i]);
            item.addActionListener(myListener);
            madlibs.add(item);
        }
        //add the "quit" option to the first file menu:
        JMenuItem item = new JMenuItem("Quit",
madlibsShortcuts[fullName.length]);
        item.addActionListener(new MenuListener());
        madlibs.add(item);
        myMenuBar.add(madlibs);

        //second menu, for the originals:
        JMenu originals = new JMenu("Originals");
        for (int i = 0; i < fullName.length; i++) {
            item = new JMenuItem(fullName[i] + " (Original)",
originalsShortcuts[i]);
            item.addActionListener(myListener);
            originals.add(item);
        }
        myMenuBar.add(originals);
        setJMenuBar(myMenuBar);
    }

    class MenuListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            //if the selection is "quit", then exit the program:
            if (e.getActionCommand().equals("Quit")) {
                System.exit(0);
            }
            //if not "quit", then see which story was selected:
            for (int i = 0; i < fullName.length; i++) {
                //check which story name matches the selected item:
                if (e.getActionCommand().indexOf(fullName[i]) == 0)
{
                    try {
                        //If the selection has "(MadLib)" in the
text, then
```

```
                        //set the storyTemplateFile to the template
for
                        //that story: (else, set it to the original
story)
                        if (e.getActionCommand().indexOf("(MadLib)")
>= 0) {
                            storyTemplateFile = shortName[i] +
"Template.txt";
                        } else {
                            storyTemplateFile = shortName[i] +
"Original.txt";
                        }
                        storyPanel.setImageIcon(new
ImageIcon("images/" + shortName[i] + ".jpg"));
                        storyTitle = fullName[i];
                        storyPanel.setTitle(storyTitle);
                        story = getStory(storyTemplateFile);
                        storyPanel.setStory(story);
                    } catch (Exception ex) {
                        System.out.println("exception: " +
ex.toString());
                        System.out.println("could not implement: " +
fullName[i]);
                    }
                }
            }

        }
    }

    public static String getStory(String templateFile) throws
FileNotFoundException {
        String story = "";
        String template = getTemplate(templateFile);
        Scanner scanner = new Scanner(template);
        int wordCount = 0; //might want to count the words

        while (scanner.hasNext()) {
            String word = scanner.next();
            wordCount++;

            //check the word against each of the tags
            for (int i = 0; i < tags.size(); i++) {
                //the new line tag is special:
                if (word.equals("<endl>")) {
                    word = "\n";
                } else {
                    //if the word matches one of the tags
                    //then get a random word from the relevant file:
                    if (word.equals("<" + tags.get(i) + ">")) {
                        word = getRandomWord(tags.get(i));
                    }
                }
            }
```

```
            story += word + " ";
        }
        scanner.close();
        return story;
    }

    public static String getTemplate(String fileName) throws
FileNotFoundException {
        //read the story template from a file:
        String template = "";
        //System.out.println("fileName = " + fileName);
        File f = new File(fileName);
        //System.out.println("file exists(): " + f.exists());
        //System.out.println("file canRead(): " + f.canRead());
        //System.out.println("file length(): " + f.length());
        //System.out.println("file canExecute: " + f.canExecute());
        //System.out.println("file getAbsoluteFile" +
f.getAbsoluteFile());

        Scanner fileScanner = new Scanner(f, "ISO-8859-1");

        //System.out.println("fileScanner.hasNext(): " +
fileScanner.hasNext());
        while (fileScanner.hasNext()) {
           // System.out.println("scanning a line of the file");
            template += fileScanner.nextLine();
        }
        fileScanner.close();
        //System.out.println("template  = " + template);
        return template;
    }

    public static String getRandomWord(String category) throws
FileNotFoundException {
        //This method gets a random word from the file
        //defined by category --> category.txt
        File file = new File(category + ".txt");
        Scanner fileScanner = new Scanner(file);
        //Now, copy all the words from the file into an array list:
        ArrayList<String> words = new ArrayList<String>();
        while (fileScanner.hasNext()) {
            String word = fileScanner.next();
            words.add(word);
            if (fileScanner.hasNext()) {
                fileScanner.nextLine();
            }
        }
        //pick a random number up to the number of words in the
list:
        int rand = (int) (words.size() * Math.random());
        //return the word at the random index:
        return words.get(rand);
    }
}
```

```
//////////////////////////////////////

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package madlib5;

import java.awt.BorderLayout;
import java.awt.Font;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

/**
 *
 * @author wwolfe3
 */
public class StoryPanel extends JPanel {

    private JLabel  imageTitle = new JLabel();
    private JTextArea jtaStory = new JTextArea();

    public StoryPanel() {
        imageTitle.setHorizontalAlignment(JLabel.CENTER);
        imageTitle.setHorizontalTextPosition(JLabel.CENTER);
        imageTitle.setVerticalTextPosition(JLabel.BOTTOM);
        imageTitle.setFont(new Font("SansSerif", Font.BOLD, 20));
        jtaStory.setFont(new Font("Serif", Font.PLAIN, 16));
        jtaStory.setLineWrap(true);
        jtaStory.setWrapStyleWord(true);
        jtaStory.setEditable(false);
        jtaStory.setSize(350, 250);

        JScrollPane scrollPane = new JScrollPane(jtaStory);
        setLayout(new BorderLayout(5,5));
        add(scrollPane, BorderLayout.CENTER);
        add(imageTitle, BorderLayout.WEST);

    }

    public void setTitle(String title) {
        imageTitle.setText(title);
    }

    public void setImageIcon(ImageIcon icon) {
        imageTitle.setIcon(icon);
    }
```

```
    public void setStory(String story) {
        jtaStory.setText(story);

    }

}
////////////////////////////////////////////
```

Marley was dead: to begin with. There is no doubt whatever about that. The register of his burial was signed by the clergyman, the clerk, the undertaker, and the chief mourner. Scrooge signed it: and Scrooge's name was good upon 'Change, for anything he chose to put his hand to. Old Marley was as dead as a door-nail. Mind! I don╒t mean to say that I know, of my own knowledge, what there is particularly dead about a door-nail. I might have been inclined, myself, to regard a coffin-nail as the deadest piece of ironmongery in the trade. But the wisdom of our ancestors is in the simile; and my unhallowed hands shall not disturb it, or the Country╒s done for. You will therefore permit me to repeat, emphatically, that Marley was as dead as a door-nail.

```
////////////////////////////////////////////
```

<name> was <stateofbeing> : to begin with. There is no <nounAbstract> whatever about that. The <randomstuff> of the <randomstuff> was signed by the <profession> , the <profession> , the <profession> , and the chief <profession> . Even Scrooge signed it: and Scrooge's name was <quality> upon <randomstuff> , for anything he chose to put a <bodypart> to. No doubt about it, old <name> was as <stateofbeing> as a <randomstuff> . Mind! I donÆt mean to say that I know, of my own knowledge, what there is particularly <dead> about a door-nail. I might have been inclined, myself, to regard a coffin-nail as the deadest piece of ironmongery in the trade. But the wisdom of our ancestors is in the simile; and my unhallowed hands shall not disturb it, or the CountryÆs done for. You will therefore permit me to repeat, emphatically, that Marley was as dead as a door-nail.

```
////////////////////////////////////////////
```

name.txt:

```
Greta
Weston
Kristan
Eilene
Quyen
Valentin
Verdie
Venice
Sandie
Kirstie
Freeda
Gita
Geraldine
```

```
...
....

//////////////////////////////////////////////////
tags.txt:

aching
adjective
adverb
amount
animal
another
bands
bendPast
bodypart
breaking
breaking
candy
causes
chemical
color
container
cool
dead
declare
dissolve
divergeP
divergePast
door
dreary
earth
ease
emotion
endl
entitle
fainting
far
flushGerund
flushVerb
force
forgotten
friend
friendship
gas
gerund
health
help
human
impel
laws
liquid
live
live
location
```

long
lookPast
loved
mankind
midnight
mutterPast
name
nature
necessary
nest
nothing
noun
nounAbstract
nounPlural
number
object
opinions
pain
pastParticiplePassive
people
poem
political
pondered
ponderPast
pourGerund
pourPast
pourVerb
powers
process
profession
quaint
quality
randomstuff
respect
roadPlural
sat
say
separation
soft
sorry
soundGerund
standPast
stateofbeing
stop
stuff
suckGerund
suckPast
suckVerb
superlative
timeofday
timeperiod
travel