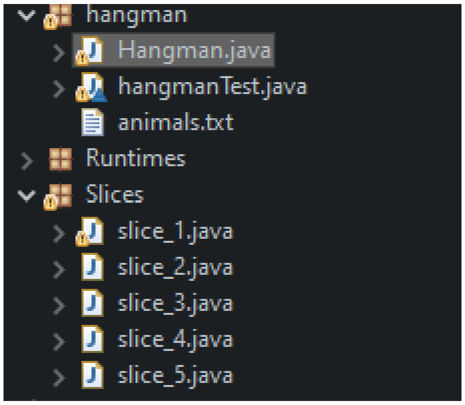
Software Quality – Assignment 2

https://github.com/JadEletry/Hangman

Static Analysis

For the static analysis portion of this task, I was able to create 5 separate slice classes that capture the functionality of my hangman program through 5 different methods that it contains while also capturing nearly the entire scope of the program itself. This technique allowed me to strip my program of its components and dial them down to their own points of interest. This clearly simplified the components, as each individual slice only focuses on its own functionality rather than having connective functionality between all methods through the main.

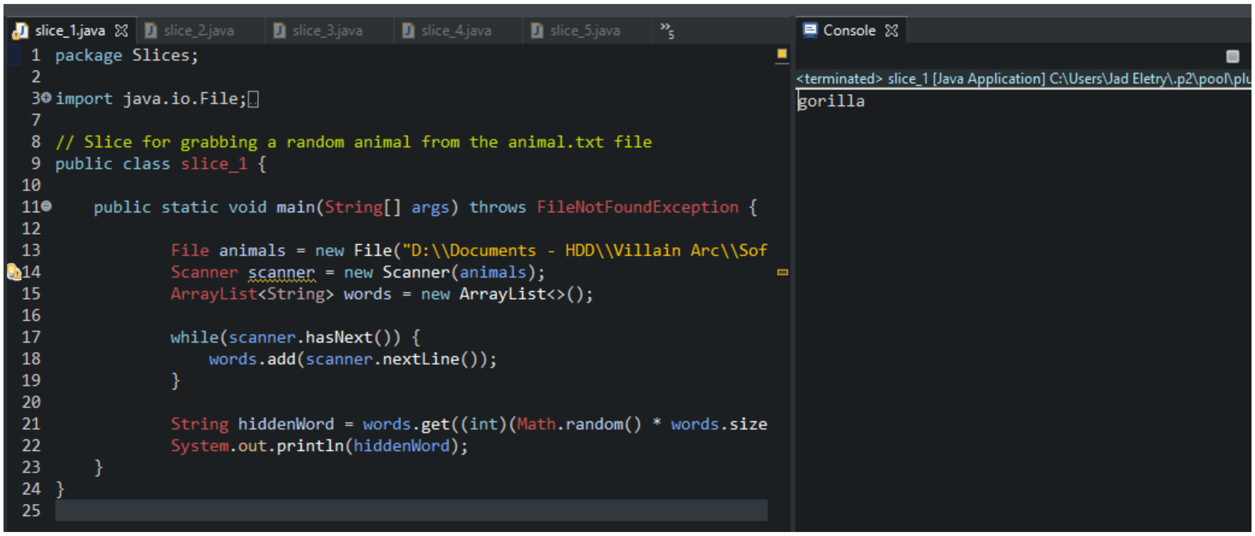


The 5 slices created were as follows:

- Slice for grabbing a random animal from the animal.txt file
- Slice for decrementing and printing remaining number of guesses left
- Slice for when the game ends
- Slice for showing the hidden word's current state
- Slice to show that the user has won

And here is each individual slice as well as their respective results:

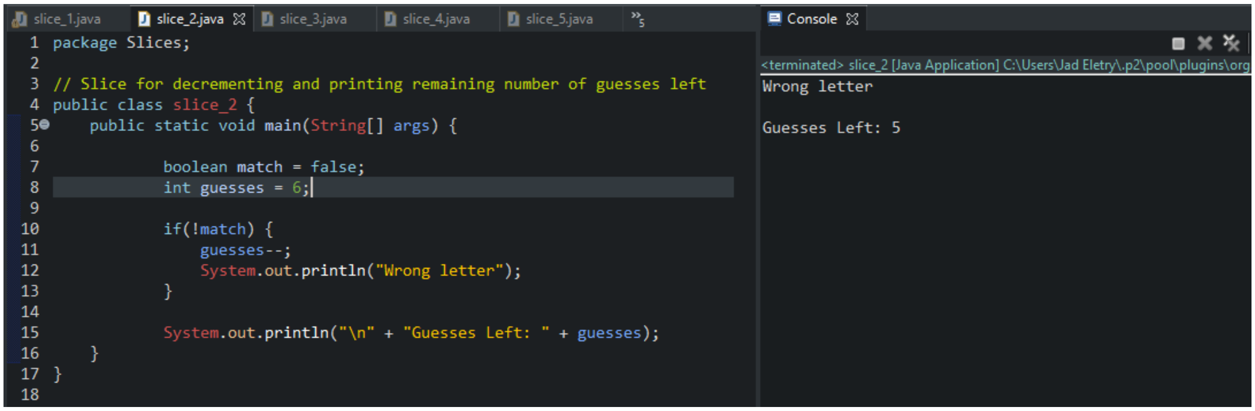
SLICE 1 & RESULT



```java
package Slices;

import java.io.File;

// Slice for grabbing a random animal from the animal.txt file
public class slice_1 {

    public static void main(String[] args) throws FileNotFoundException {

            File animals = new File("D:\\Documents - HDD\\Villain Arc\\Sof
            Scanner scanner = new Scanner(animals);
            ArrayList<String> words = new ArrayList<>();

            while(scanner.hasNext()) {
                words.add(scanner.nextLine());
            }

            String hiddenWord = words.get((int)(Math.random() * words.size
            System.out.println(hiddenWord);
    }
}
```

```
<terminated> slice_1 [Java Application] C:\Users\Jad Eletry\.p2\pool\plu
gorilla
```

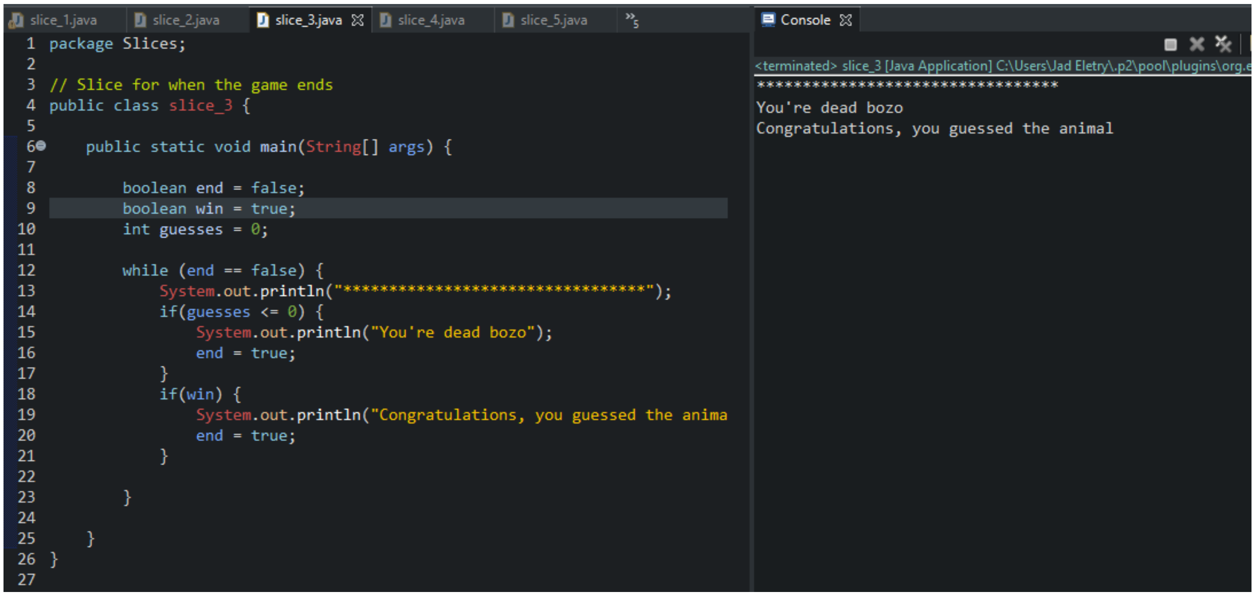
SLICE 2 & RESULT



```java
package Slices;

// Slice for decrementing and printing remaining number of guesses left
public class slice_2 {
    public static void main(String[] args) {

            boolean match = false;
            int guesses = 6;

            if(!match) {
                guesses--;
                System.out.println("Wrong letter");
            }

            System.out.println("\n" + "Guesses Left: " + guesses);
    }
}
```

```
<terminated> slice_2 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org
Wrong letter

Guesses Left: 5
```

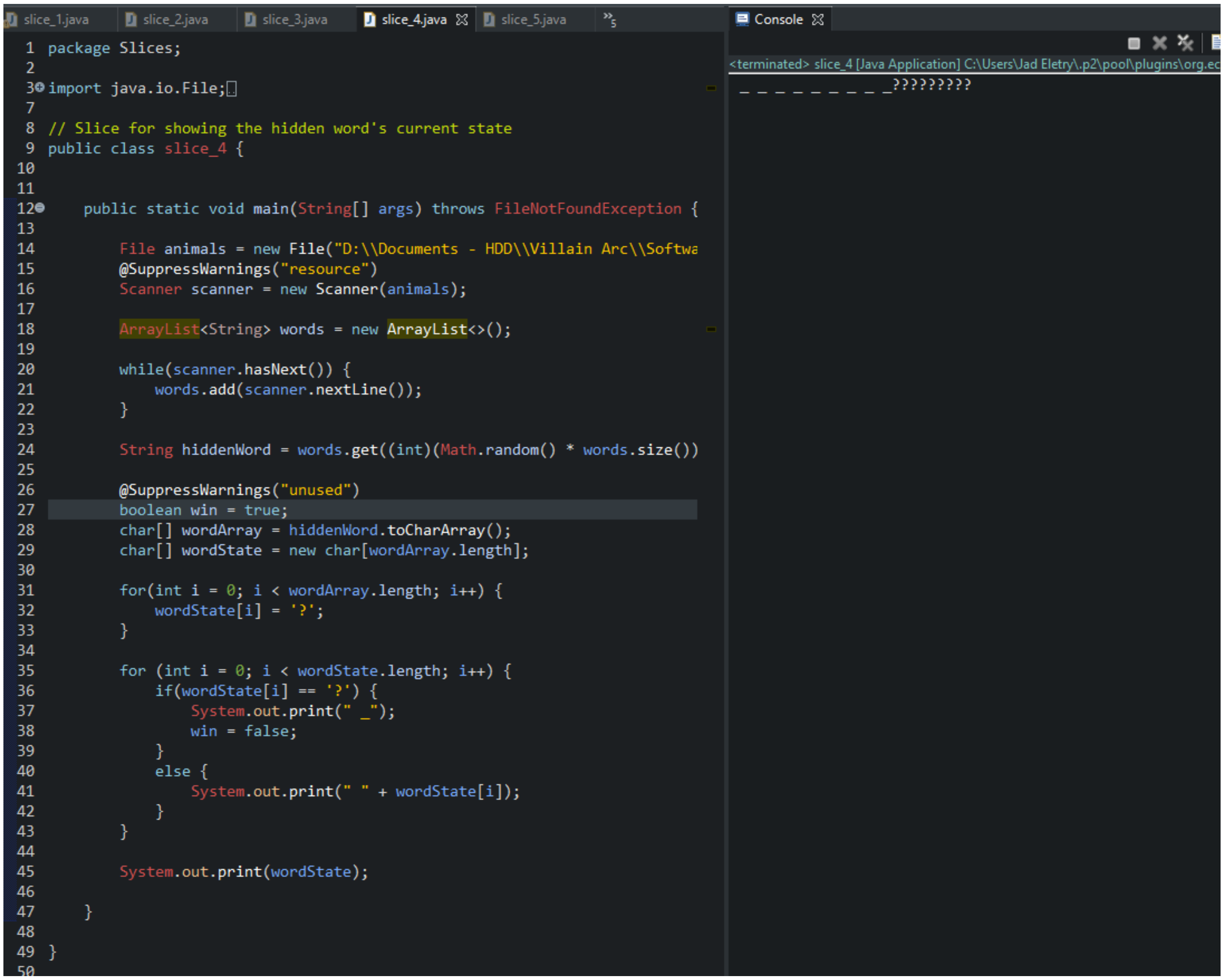
SLICE 3 & RESULT



```java
package Slices;

// Slice for when the game ends
public class slice_3 {

    public static void main(String[] args) {

        boolean end = false;
        boolean win = true;
        int guesses = 0;

        while (end == false) {
            System.out.println("*********************************");
            if(guesses <= 0) {
                System.out.println("You're dead bozo");
                end = true;
            }
            if(win) {
                System.out.println("Congratulations, you guessed the anima
                end = true;
            }

        }

    }
}

```

```
<terminated> slice_3 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.e
*********************************
You're dead bozo
Congratulations, you guessed the animal
```

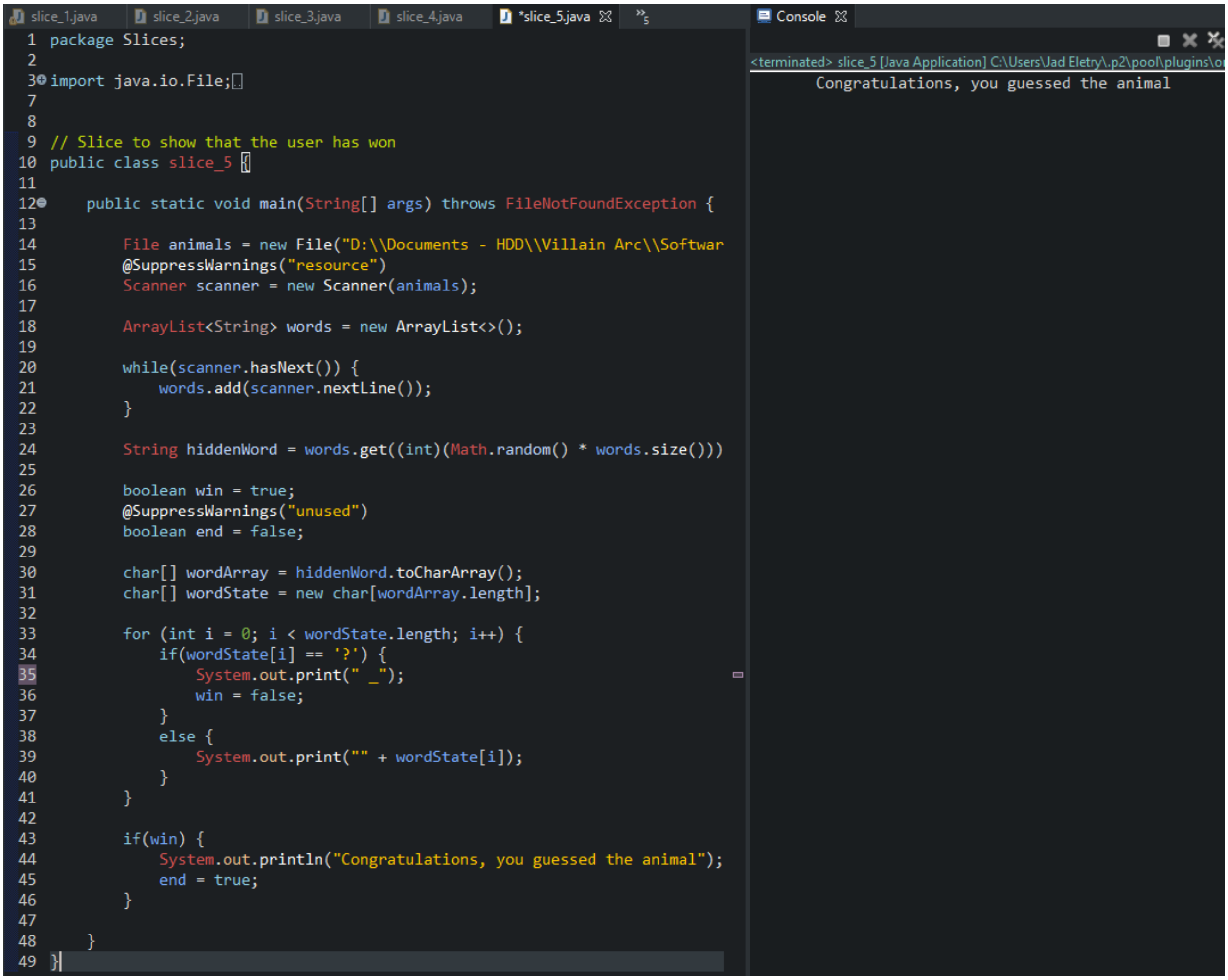
SLICE 4 & RESULT



```java
package Slices;

import java.io.File;

// Slice for showing the hidden word's current state
public class slice_4 {


    public static void main(String[] args) throws FileNotFoundException {

        File animals = new File("D:\\Documents - HDD\\Villain Arc\\Softwa
        @SuppressWarnings("resource")
        Scanner scanner = new Scanner(animals);

        ArrayList<String> words = new ArrayList<>();

        while(scanner.hasNext()) {
            words.add(scanner.nextLine());
        }

        String hiddenWord = words.get((int)(Math.random() * words.size())

        @SuppressWarnings("unused")
        boolean win = true;
        char[] wordArray = hiddenWord.toCharArray();
        char[] wordState = new char[wordArray.length];

        for(int i = 0; i < wordArray.length; i++) {
            wordState[i] = '?';
        }

        for (int i = 0; i < wordState.length; i++) {
            if(wordState[i] == '?') {
                System.out.print(" _");
                win = false;
            }
            else {
                System.out.print(" " + wordState[i]);
            }
        }

        System.out.print(wordState);

    }

}

```

```
<terminated> slice_4 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.ec
 _ _ _ _ _ _ _ _ _?????????
```

SLICE 5 & RESULT

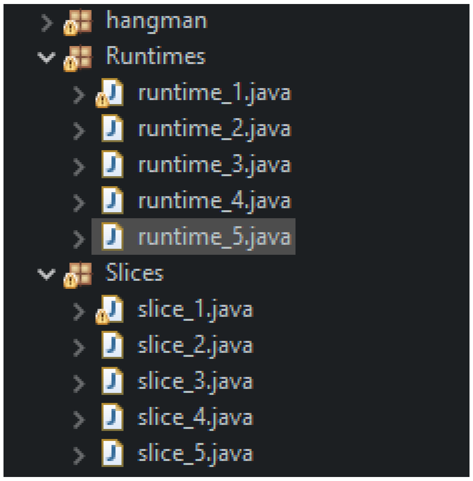


```java
package Slices;

import java.io.File;


// Slice to show that the user has won
public class slice_5 {

    public static void main(String[] args) throws FileNotFoundException {

        File animals = new File("D:\\Documents - HDD\\Villain Arc\\Softwar
        @SuppressWarnings("resource")
        Scanner scanner = new Scanner(animals);

        ArrayList<String> words = new ArrayList<>();

        while(scanner.hasNext()) {
            words.add(scanner.nextLine());
        }

        String hiddenWord = words.get((int)(Math.random() * words.size()))

        boolean win = true;
        @SuppressWarnings("unused")
        boolean end = false;

        char[] wordArray = hiddenWord.toCharArray();
        char[] wordState = new char[wordArray.length];

        for (int i = 0; i < wordState.length; i++) {
            if(wordState[i] == '?') {
                System.out.print(" _");
                win = false;
            }
            else {
                System.out.print("" + wordState[i]);
            }
        }

        if(win) {
            System.out.println("Congratulations, you guessed the animal");
            end = true;
        }

    }
}
```

```
<terminated> slice_5 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\o
       Congratulations, you guessed the animal
```

Dynamic Analysis

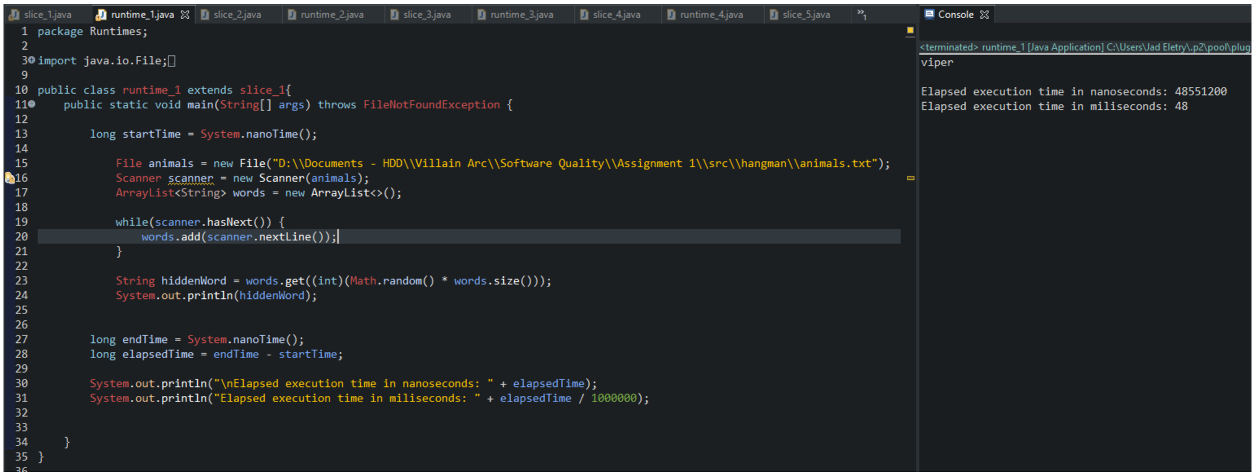
For dynamic analysis, we used runtime instrumentation which allowed us to test and determine the total elapsed time of each individual slice from the previous task. The elapsed time was calculated in both nanoseconds and milliseconds. When we calculate for runtime, we're able to observe the efficiency of our program and since we've created slices to focus on individual components of our main program, we're able to see which component is the least and most efficient with respect to their algorithms. Generally, you want faster run times as this means the complexity of your program is at a minimum, which is what depicts the good programs from the bad programs. I've created a separate package to show all slices and their respective runtimes.



```
> hangman
v Runtimes
    > runtime_1.java
    > runtime_2.java
    > runtime_3.java
    > runtime_4.java
    > runtime_5.java
v Slices
    > slice_1.java
    > slice_2.java
    > slice_3.java
    > slice_4.java
    > slice_5.java
```

And here are each slice's runtimes:

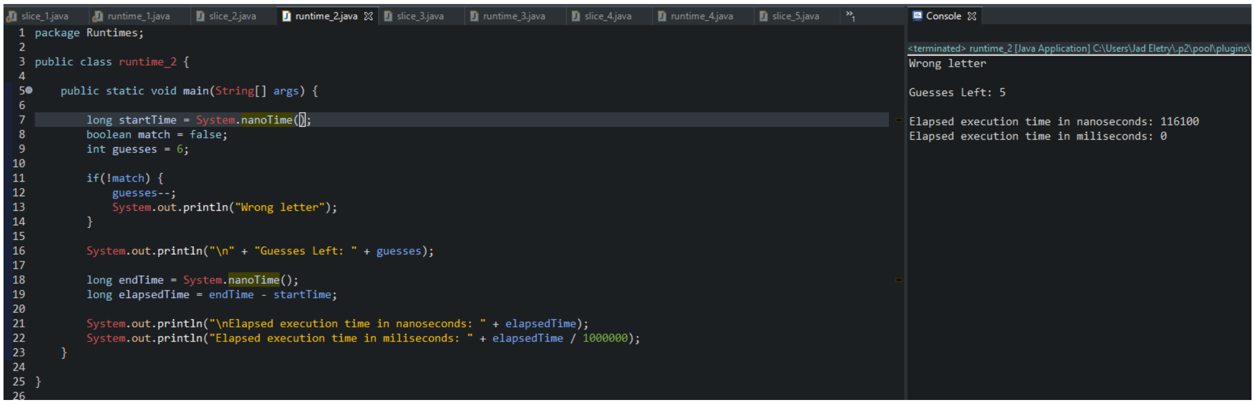
SLICE 1 & RUNTIME



```java
package Runtimes;

import java.io.File;

public class runtime_1 extends slice_1{
    public static void main(String[] args) throws FileNotFoundException {

        long startTime = System.nanoTime();

            File animals = new File("D:\\Documents - HDD\\Villain Arc\\Software Quality\\Assignment 1\\src\\hangman\\animals.txt");
            Scanner scanner = new Scanner(animals);
            ArrayList<String> words = new ArrayList<>();

            while(scanner.hasNext()) {
                words.add(scanner.nextLine());
            }

            String hiddenWord = words.get((int)(Math.random() * words.size()));
            System.out.println(hiddenWord);


        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;

        System.out.println("\nElapsed execution time in nanoseconds: " + elapsedTime);
        System.out.println("Elapsed execution time in miliseconds: " + elapsedTime / 1000000);


    }
}
```

```
<terminated> runtime_1 [Java Application] C:\Users\Jad Eletry\.p2\pool\plug
viper

Elapsed execution time in nanoseconds: 48551200
Elapsed execution time in miliseconds: 48
```

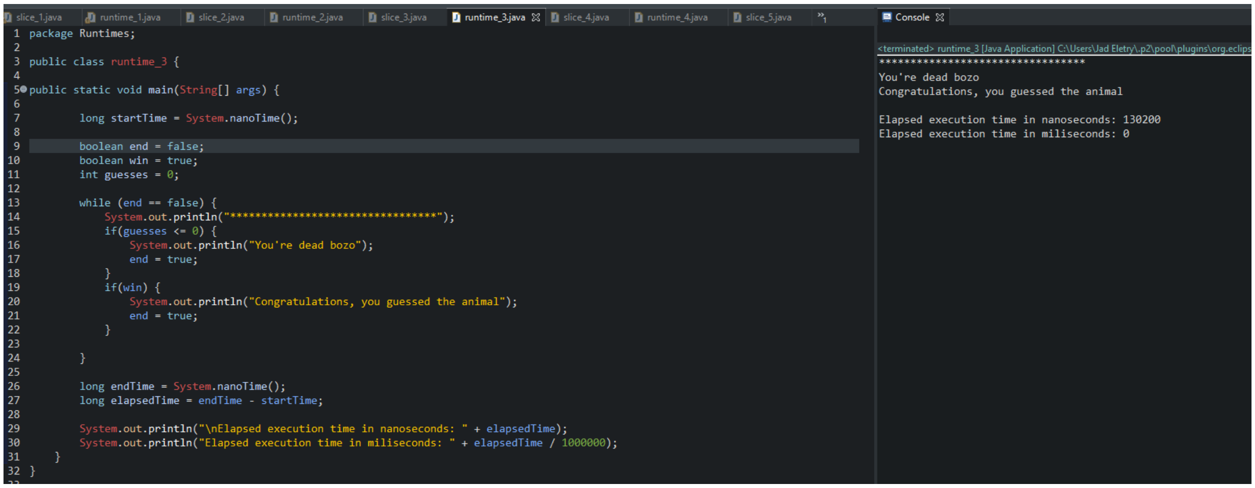
SLICE 2 & RUNTIME



```java
package Runtimes;

public class runtime_2 {

    public static void main(String[] args) {

        long startTime = System.nanoTime();
        boolean match = false;
        int guesses = 6;

        if(!match) {
            guesses--;
            System.out.println("Wrong letter");
        }

        System.out.println("\n" + "Guesses Left: " + guesses);

        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;

        System.out.println("\nElapsed execution time in nanoseconds: " + elapsedTime);
        System.out.println("Elapsed execution time in miliseconds: " + elapsedTime / 1000000);
    }

}
```

```
<terminated> runtime_2 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\
Wrong letter

Guesses Left: 5

Elapsed execution time in nanoseconds: 116100
Elapsed execution time in miliseconds: 0
```

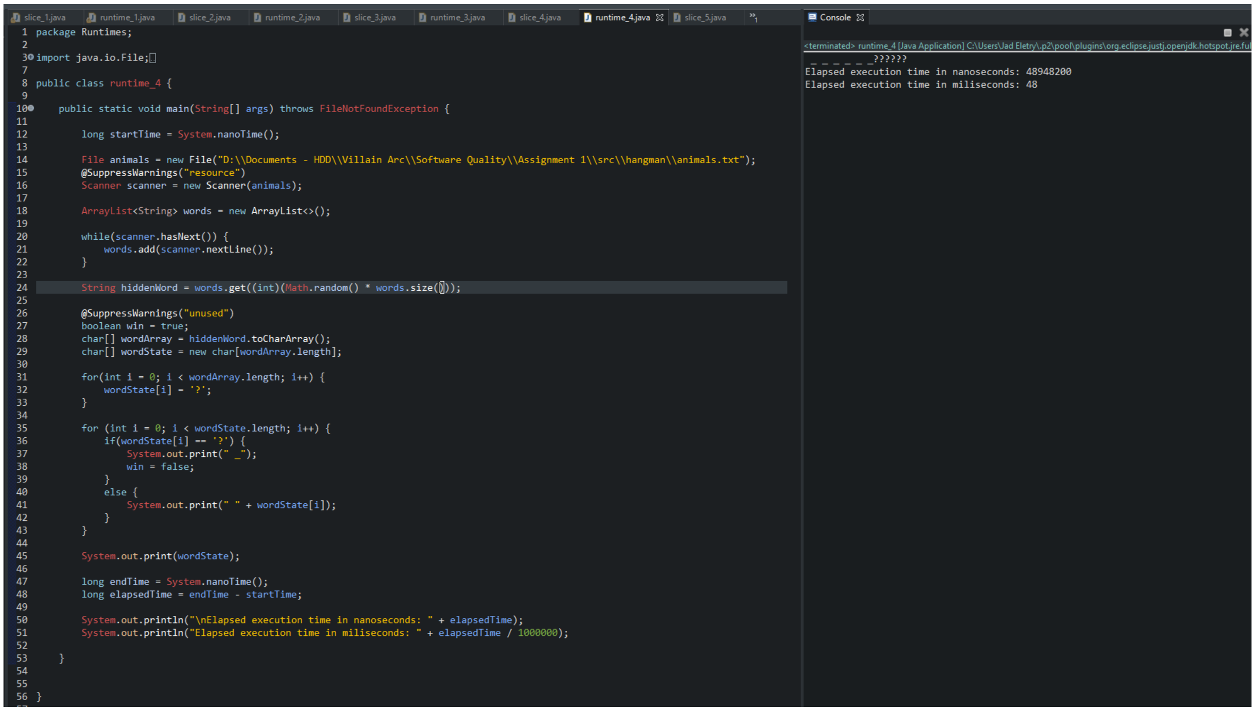
SLICE 3 & RUNTIME



```java
package Runtimes;

public class runtime_3 {

public static void main(String[] args) {

        long startTime = System.nanoTime();

        boolean end = false;
        boolean win = true;
        int guesses = 0;

        while (end == false) {
            System.out.println("**********************************");
            if(guesses <= 0) {
                System.out.println("You're dead bozo");
                end = true;
            }
            if(win) {
                System.out.println("Congratulations, you guessed the animal");
                end = true;
            }

        }

        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;

        System.out.println("\nElapsed execution time in nanoseconds: " + elapsedTime);
        System.out.println("Elapsed execution time in miliseconds: " + elapsedTime / 1000000);
    }
}
```

```
<terminated> runtime_3 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.eclips
**********************************
You're dead bozo
Congratulations, you guessed the animal

Elapsed execution time in nanoseconds: 130200
Elapsed execution time in miliseconds: 0
```

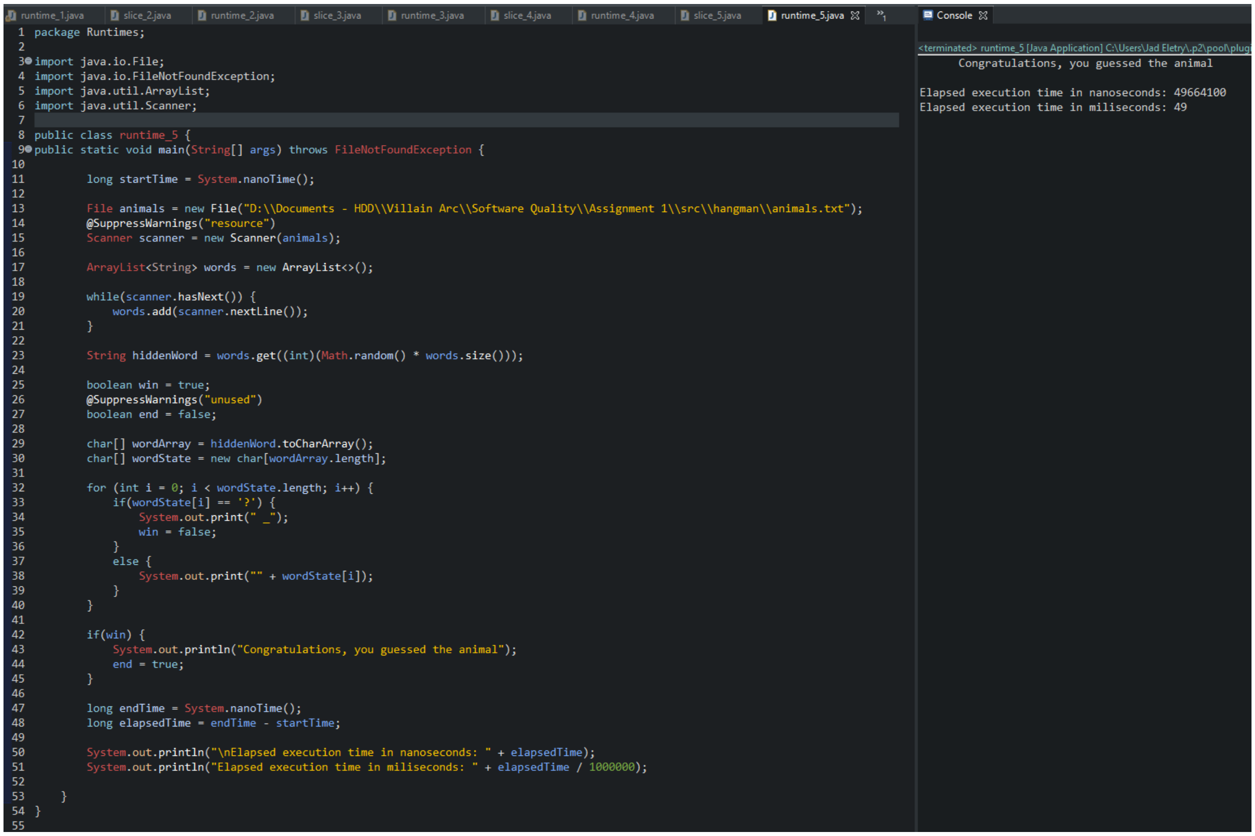
SLICE 4 & RUNTIME



```java
package Runtimes;

import java.io.File;

public class runtime_4 {

    public static void main(String[] args) throws FileNotFoundException {

        long startTime = System.nanoTime();

        File animals = new File("D:\\Documents - HDD\\Villain Arc\\Software Quality\\Assignment 1\\src\\hangman\\animals.txt");
        @SuppressWarnings("resource")
        Scanner scanner = new Scanner(animals);

        ArrayList<String> words = new ArrayList<>();

        while(scanner.hasNext()) {
            words.add(scanner.nextLine());
        }

        String hiddenWord = words.get((int)(Math.random() * words.size()));

        @SuppressWarnings("unused")
        boolean win = true;
        char[] wordArray = hiddenWord.toCharArray();
        char[] wordState = new char[wordArray.length];

        for(int i = 0; i < wordArray.length; i++) {
            wordState[i] = '?';
        }

        for (int i = 0; i < wordState.length; i++) {
            if(wordState[i] == '?') {
                System.out.print(" _");
                win = false;
            }
            else {
                System.out.print(" " + wordState[i]);
            }
        }

        System.out.print(wordState);

        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;

        System.out.println("\nElapsed execution time in nanoseconds: " + elapsedTime);
        System.out.println("Elapsed execution time in miliseconds: " + elapsedTime / 1000000);

    }


}
```

```
<terminated> runtime_4 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
 _ _ _ _ _ _??????
Elapsed execution time in nanoseconds: 48948200
Elapsed execution time in miliseconds: 48
```

SLICE 5 & RUNTIME



```java
package Runtimes;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class runtime_5 {
public static void main(String[] args) throws FileNotFoundException {

        long startTime = System.nanoTime();

        File animals = new File("D:\\Documents - HDD\\Villain Arc\\Software Quality\\Assignment 1\\src\\hangman\\animals.txt");
        @SuppressWarnings("resource")
        Scanner scanner = new Scanner(animals);

        ArrayList<String> words = new ArrayList<>();

        while(scanner.hasNext()) {
            words.add(scanner.nextLine());
        }

        String hiddenWord = words.get((int)(Math.random() * words.size()));

        boolean win = true;
        @SuppressWarnings("unused")
        boolean end = false;

        char[] wordArray = hiddenWord.toCharArray();
        char[] wordState = new char[wordArray.length];

        for (int i = 0; i < wordState.length; i++) {
            if(wordState[i] == '?') {
                System.out.print(" _");
                win = false;
            }
            else {
                System.out.print("" + wordState[i]);
            }
        }

        if(win) {
            System.out.println("Congratulations, you guessed the animal");
            end = true;
        }

        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;

        System.out.println("\nElapsed execution time in nanoseconds: " + elapsedTime);
        System.out.println("Elapsed execution time in miliseconds: " + elapsedTime / 1000000);

    }
}
```

```
<terminated> runtime_5 [Java Application] C:\Users\Jad Eletry\.p2\pool\plugi
      Congratulations, you guessed the animal

Elapsed execution time in nanoseconds: 49664100
Elapsed execution time in miliseconds: 49
```

Now we can see which component(s) are the most efficient (listed below from most to least efficient) → *in nanoseconds & milliseconds*

- Slice 2 with 116100ns & 0ms
- Slice 3 with 130200ns & 0ms
- Slice 1 with 48551200ns & 48ms
- Slice 4 with 48948200ns & 48ms
- Slice 5 with 49664100ns & 49ms

As we can see, dynamic analysis allows us to probe our program and we can clearly see that slices or components 1, 4, & 5 are the least efficient meaning in the real world this would need to be looked at and improved for runtime and complexity.

Challenges

Some challenges I faced while working was for one, trying to implement automated slicing in a program. Although this was my ideal plan and what I originally wanted to do, it was very hard to understand and try and wrap my head around while trying to use it to parse m program as the variables that I include in my program are distributed quite well across different components within the program. Instead, I manually parsed my program for the specific variables that I included and ensured each slice was executable. Manually parsing allowed me to understand how each component works within my program and how they execute without referencing other variables.