Assignment 1 Cathal Lawlor 21325456

Problem Statement:

The problem is that we are to design a program that uses (preferably) one array, holding the alphabet. You must time how long the user takes from commencing the game all the way to them iterating through the array. Throughout this you let the user know what input to type and let them know when they've finished, printing their time taken.

Analysis

We are going to use / have:

- An array of the alphabet.
- A general method letting the user know how to proceed, calling any other necessary methods.
- A class / method for timing.
- Scanners to take in user input.
- Input sanitising and error checking.
- Methods to iterate through the array forwards or backwards.
- If correct letter is typed these methods proceed, if not we let the user try again.

I decided on using a scanner to take user input for deciding between forwards/backwards or to exit. There is a case statement to complete this using the String direction.

There is a timer class that returns a long for milliseconds at start and then takes the long back in for its end method. This method produces the seconds taken and prints to the screen.

The user decided the direction, where either of the methods iterateForwards or iterateBackwards is called alongside the timer starting. If direction isn't chosen, then system will exit.

These methods both iterate along the length of the array, in their chosen direction with a for loop.

In this for loop there is a while checking if the input c from user is the correct element in the array. Repeatedly scanning until condition is met.

The system will print out the next required letter, and then iterate to the next element in the alphabet array or finish the array.

If the array is finished, we congratulate the user, stop the timer and print the time taken.

Testing: I will test if the program doesn't accept wrong chars or strings, if it can do the arrays both forwards and backwards and if it exits if user doesn't decide a direction.

Methods used:

Main – instantiates a game session for the user.

inputFIeld – method called to let user decide direction of the game.

iterateForwards & iterateBackwards – methods to iterate through the alphabet in chosen direction.

timerCalculationStart – method to note and return the time method is called at in milliseconds.

timerCalculationEnd – method to end the timer and to calculate and print the given time taken.

Testing:

```
Game started, you'll be timed after this input!.
Run through alphabet forwards or backwards? Enter F or B:
f

All inputs must be lower case, hit enter between letters

Please type: a
a
Well done, now type: b
b
Well done, now type: c
c
Well done, now type: d
d
Well done, now type: e
e
Well done, now type: f
gfs
a
f
Well done, now type: g
g
Well done, now type: h
h
Well done, now type: i
i
Well done, now type: j
j
Well done, now type: k
k
Well done, now type: l
l
Well done, now type: m
m
Well done, now type: n
n
Well done, now type: o
o
```

Forwards & continued (including wrong letters in sequence or string): Expected output to not accept wrong chars or strings.

```
Well done, now type: p
p
Well done, now type: q
q
Well done, now type: r
r
Well done, now type: s
s
Well done, now type: t
t
Well done, now type: u
u
Well done, now type: v
v
Well done, now type: w
w
Well done, now type: x
x
Well done, now type: y
y
Well done, now type: z
z

You completed it, well done!
Time taken to complete: 18 seconds
```

Backwards & continued (including wrong letters in sequence or string): Expected output to not take wrong chars or strings.

```
Game started, you'll be timed after this input!.
Run through alphabet forwards or backwards? Enter F or B:
b

All inputs must be lower case,
Please type: z
z
Well done, now type: y
y
Well done, now type: x
x
Well done, now type: w
w
Well done, now type: v
r
t
v
Well done, now type: u
u
Well done, now type: t
t
Well done, now type: s
s
Well done, now type: r
r
Well done, now type: q
q
Well done, now type: p
p
Well done, now type: o
o
Well done, now type: n
n
Well done, now type: m
m
Well done, now type: l
l
Well done, now type: k
k
```

```
Well done, now type: k
k
Well done, now type: j
j
Well done, now type: i
i
Well done, now type: h
h
Well done, now type: g
g
Well done, now type: f
f
Well done, now type: e
e
Well done, now type: d
d
Well done, now type: c
c
Well done, now type: b
b
Well done, now type: a
a

You completed it, well done!
Time taken to complete: 24 seconds
```

Entering something that is F or B: Expected output to exit program.

```
Game started, you'll be timed after this input!.
Run through alphabet forwards or backwards? Enter F or B:
df
Invalid, must be F or B to start.
Good luck - try again later!
```

Code:

alphabetRead.java

```
public class alphabetRead {

    public static void main(String[] args) {
        userCharInput gameSession = new userCharInput(); //creating a game instance
        gameSession.inputField(); //starting game for use
    }
}
```

timings.java

```
public class timings {

    public long timeCalculationStart() {
        long startTime = System.currentTimeMillis(); //setting the starting time
        return startTime; //returning the time program started at
    }

    public void timeCalculationEnd(Long startTime) {
        long timeElapsed = System.currentTimeMillis() - startTime;//system time minus the
previous start time
        long totalElapsedSeconds = timeElapsed / 1000; //milliseconds into seconds

        //letting user know total time taken (in seconds, could easily implement milli if
required using the modulous of totalElapsedSeconds)
        System.out.println("Time taken to complete: " + totalElapsedSeconds + " seconds");
    }


}
```

userCharInput.java

```
import java.util.Scanner;  // Import the Scanner class

public class userCharInput {

    static char alphabetArray[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
                                   'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
                                   's', 't', 'u', 'v', 'w', 'x', 'y', 'z'}; //alphabet array :)
    long startTime = 0;
    timings timer = new timings();  //creating timing object


    public void inputField() {


        try (Scanner myObj = new Scanner(System.in)) {
            System.out.println("\nGame started, you'll be timed after this input!.");
            System.out.println("Run through alphabet forwards or backwards? Enter F or B: ");

            String direction = myObj.nextLine();  // Read user input


            direction = direction.toUpperCase(); //sanatising input, in case the user is lazy -
like me!
            startTime = timer.timeCalculationStart(); //calling timings to start

            switch(direction) {
                case "F": //if F we have a forwards game method called
                    iterateForwards();
                    break;

                case "B": //likewise, B calls a backwards method
                    iterateBackwards();
                    break;

                default: //else let user know input is invalid - wish them a good day
                    System.out.println("Invalid, must be F or B to start. \nGood luck - try
again later!");
                    System.exit(0);
            }

        } catch (Exception e) {
            System.exit(0);
        }

    }

    public void iterateForwards() {
        try (Scanner charObjScanner = new Scanner(System.in)) { //try a scanner
            char c = '.'; //initialise char c.

            System.out.println("\nAll inputs must be lower case, hit enter between letters");
```

```
            System.out.println("\nPlease type: " + alphabetArray[0]); //letting user know what
to start off with

            for (int i = 0; i < alphabetArray.length; i++) { // for length of alphabet we'll
iterate across it
                while (!(alphabetArray[i] == c)) { //while c is not the same as current array
element
                    c = charObjScanner.next().charAt(0); //we scan a new character in
                    //if it was charObjScanner.next(".").charAt(0), it would crash the program
when more than one char entered
                }
                if (i < alphabetArray.length - 1){ //if statement to make sure we don't access
array elements that don't exist
                System.out.println("Well done, now type: " + alphabetArray[i + 1]);
                }
            }
            System.out.println("\nYou completed it, well done!");

            timer.timeCalculationEnd(startTime);  //once game is finished we stop the timer
            //it also outputs the total time to user
        }
        catch (Exception e) { System.exit(0); } //catching errors with try and catch
    }

    public void iterateBackwards() {
        try (Scanner charObjScanner = new Scanner(System.in)) { //same as above, new scanner
and char
            char c = '.';

            System.out.println("\nAll inputs must be lower case, \nPlease type: " +
alphabetArray[alphabetArray.length - 1]); //type last element of array to start

            for (int i = alphabetArray.length - 1; i >= 0; i--) { //starting at end of array,
we'll descend across it to the start.
                while (!(alphabetArray[i] == c)) {
                    c = charObjScanner.next().charAt(0);
                }
                if (i > 0){ //making sure not to go past the index of array, we're not in
python so we can go -1 on arrays
                    System.out.println("Well done, now type: " + alphabetArray[i - 1]);
                }
            }

            System.out.println("\nYou completed it, well done!");

            timer.timeCalculationEnd(startTime);  //once game is finished we stop the timer
            //it also outputs the total time to user
        }
        catch (Exception e) { System.exit(0);  } //catch again
    }
}
```