## Labs

| (2) Level A | (1) Level B | (1) Level C |
|---|---|---|
| ArrayList Data Set | Hangman | Mouse Trap |
| Random Seating Chart | | |

# ArrayList Data Set

Write a program that will allow the user to manipulate an ArrayList of Integer values.

The program will need to have a looping menu with the following options: add values, remove a value, remove a location, insert a value, clear list, get size, get index, print, total, average and exit.

**Menu Options:**

- Add values – add values until 0 is entered
  - Each value is added to the end of the ArrayList.
  - Zero is not added.
- Change a value –Changes the value at a given location.
  - Prints an error if the index is invalid.
- Insert a value – Adds a single value at a given location.
  - Prints an error if the index is invalid.
- Get Index – Prints the value at a given location.
  - Prints an error if the index is invalid.
- Get Size – Prints the number of elements in the ArrayList
- Print – Displays all the values in the ArrayList
- Total – Prints the total of all the values in the ArrayList
- Average – Prints the average of all the values in the ArrayList
- Remove a value – Attempts to remove a value given.
  - If the remove was successful print “remove successful”
  - If the remove was unsuccesful print “value not found”.
- Remove a location – Removes the value at a given location and prints the value that was removed.
  - Prints an error if the index is invalid.
- Clear –Empties the ArrayList
- Exit – Exits the program

Rubric (counts as 1 other grade)

| Points | Task |
|---|---|
| 10 | Menu |
| 5 | Add values |
| 10 | Change value |
| 10 | Insert |
| 10 | Get size |
| 5 | Print |
| 5 | Total |
| 5 | Average |
| 10 | Remove value |
| 10 | Remove location |
| 5 | Clear |
| 5 | Exit |
| 10 | Formatting and neatness |

# Random Seating Chart

Write a program that will gather student names from the user until “Done” is entered. The program will then ask the user for the number of seats in the classroom.

When the number of seats is less than the number of students the program will display “Error not enough seats in your class”.

When the number of seats is greater than or equal to the number of students the program will assign each student a seat randomly.

**Data Storage:**

- Names – ArrayList of Strings
- Seats – ArrayList of Integers

## Rubric (counts as 1 other grade)

| Points | Task |
|---|---|
| 20 | Gathers student’s names |
| 20 | Gets number of seats |
| 20 | Error when there are not enough seats |
| 20 | Correctly assigns seats |
| 20 | Formatting and neatness |

# Hangman

## Directions

Create a program that allows the user to play hangman.

Before each guess the user will see what the sentence looks like based on the current letters they have guessed, the guessed vowels, the guessed consonants and the number of guesses that the user has before the game ends.

The game will end once the user has run out of guesses or has guessed all the letters in the sentence correctly.

The sentence will be randomly selected from 10 sentences you create.

**Data storage for your program:**

- **10 Sentences – ArrayList of Strings**
- **Guessed Vowels – ArrayList of Characters**
- **Guessed Consonants – ArrayList of Characters**
- **char[] sentence – Stores the sentence**

## Note:

A string can be turned into a character array with the toCharArray method.

**Example:**

```
String s = "Billy is hungry."
char[] sentence = s.toCharArray();
```

## Extra Credit

Instead of telling the user how many more letters they can miss before they lose, draw what the hangman and stage would look like.

## Rubric (counts as 1 minor grade)

| Points | Task |
|---|---|
| 10 | Random sentence |
| 20 | Display of sentence with non-guessed letters hidden |
| 10 | Guessed vowels |
| 10 | Guessed consonants |
| 10 | Guesses left |
| 10 | Losing |
| 10 | Winning |
| 20 | Formatting and neatness |
| 15 | Extra credit |

# Mouse Trap

Write a program that allows the user to attempt traping a mouse in a 9x9 grid of chars by erecting up to six temporary walls.

The location of the walls will be stored in an ArrayList of Points. The array will never be longer than 6 elements. When the size of the ArrayList reaches 7 it will remove the oldest wall.

The user wins when the mouse is trapped. The mouse is trapped when it can no longer move forward, backward, left or right. At the end of the game the user will be told the number of moves it took them to win.

The mouse has a 50% to move forward, a 20% chance to move left and a 20% chance to move right and 10% chance to move backwards (based on current direction).

Each turn the mouse will move if possible. This means that if it chooses a blocked direction it will chose a new direction.

The mouse will always start facing up in square (4, 4).

**Grid Symbols:**

^ - mouse facing up
v – mouse facing down
< - mouse facing left
> – mouse facing right
W – wall
- – empty

**Example maze:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | W | W | - | - | - |
| - | - | - | - | ^ | W | - | - | - |
| - | - | - | - | W | - | W | - | - |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |

Rubric (counts as 2 minor grades)

| Points | Task |
|---|---|
| 20 | Mouse moves properly |
| 20 | Walls can be added |
| 20 | Walls decay |
| 20 | Ends when mouse cannot move |
| 10 | Correct turn count |
| 10 | Formatting and neatness |