SCC.111: DIVING INTO JAVA PROGRAMMING

This task aims to introduce to Java programming. You will be porting your C++ Dice program from Week 12 to Java. You will also be working on a simple graphics library to create a simple game. You will be working in groups for the second part of the task. Through this task you will learn to:

- ☐ Implement basic code in Java.
- ☐ Compile and run Java code.
- ☐ Create simple graphics using Java.

TASK 1: PORTING THE DICE TO JAVA

The first task is simple and short, you shouldn't spend more than **10 minutes** before moving to the second part. If you are stuck, ask a teaching staff to help you out.

Goals

This lab aims to get you familiar with Java syntax. Your goal in this task is to port (convert) your C++ Dice program from Week 12 to Java. You will create a `Dice` class and a `Player` class in Java. You are given a `Driver` class to test your code.

1. Create three *.java* files: `Dice.java` , `Player.java` , and `Driver.java`
2. Put your C++ *Dice* class and *Player* class and their functions in `Dice.java` and `Player.java` , respectively. Then modify the C++ code until it becomes Java code. You can either use your own week 12 implementation or the following C++ code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

class Dice {
public:
```

```
    int numSides;
    int roll();
};

int Dice::roll() {
    return rand() % numSides + 1;
}

class Player {
public:
    char *name;
    int makeGuess();
};

int Player::makeGuess() {
    int guess;
    printf("%s, make a guess (between 1 and 6): ", name);
    scanf("%d", &guess);
    return guess;
}

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));

    Dice sixSidedDice;
    sixSidedDice.numSides = 6;

    Player player1;
    player1.name = "Player 1";

    int guess = player1.makeGuess();
```

```
    int rollResult = sixSidedDice.roll();

    if (guess == rollResult) {
        printf("Congratulations! %d is correct. You win!\n", guess);
    } else {
        printf("Sorry, %d is incorrect. The correct answer is %d. Try again!\n",
↪   guess, rollResult);
    }

    return 0;
}
```

3. Test your code in the `Driver.java` . You can use the following code.

```
public class Driver {
    public static void main(String[] args) {
        testGame("Player 1", 6);
        testGame("Player 2", 4);
    }

    private static void testGame(String playerName, int numSides) {
        Dice sixSidedDice = new Dice();
        sixSidedDice.numSides = numSides;

        Player player = new Player();
        player.name = playerName;

        int guess = player.makeGuess();
        int rollResult = sixSidedDice.roll();

        displayResult(playerName, guess, rollResult);
```

```
    }

    private static void displayResult(String playerName, int guess, int rollResult) {
        if (guess == rollResult) {
            System.out.printf("Congratulations, %s! %d is correct. You win!\n",
            ↪  playerName, guess);
        } else {
            System.out.printf("Sorry, %s. %d is incorrect. The correct answer is %d.
            ↪  Try again!\n",
                    playerName, guess, rollResult);
        }
    }
}
```

TIPS

1. Refer to the **last slides** of Week 15 Lecture 1 for Java code to **generate random numbers**.
2. Check out this **cheatsheet** for *Java* **syntax equivalents** to *C* and *C++*.
3. In the terminal, navigate to your directory `cd path/to/your/java/files` . Use `javac *.java` to compile all the Java files in the directory. Use `java Driver` to run and test your Code.

TASK 2: GOODIES AND BADDIES

Note: **FOR THIS EXERCISE WE RECOMMEND YOU WORK IN SMALL GROUPS. WE'LL ALSO CONTINUE WORKING ON THIS TASK IN NEXT WEEK'S LAB SESSION, SO DON'T EXPECT TO FINISH THIS WEEK!**

You all seemed to enjoy working created simple graphics and animations in Week 14… So I thought we could continue the theme and explore Object-Oriented (OO) programming through a slightly more advanced set of classes… this time using the *Java* programming language.

We have written a simple set up classes for you, which will allow you to create windows and draw Lines, Rectangles, Balls and Text. This time the interface is entirely written in Java, a little higher performance,

and can be interactive… but still very object-oriented.

Download the **GameArena.zip** file from moodle. In there you will find a set of Java classes that you are required to create these simple graphics. Extract this zip file to somewhere on your H drive. We recommend you write your code in the same directory as the GameArena Java files for simplicity. Note also that full Javadoc API documentation has also been provided in the doc folder. Open the "index.html" file in your browser to access this.

Getting Started

Review that you understand the capabilities of the classes you have been given. Try to learn how to use the classes from the API documentation. Whilst you are free to read the code for the GameArena classes, you do not need to do so to complete this task. **You should not need to change any of the GameArena classes**.

Here is a very simple program that will draw a single ball on the screen, to help you get started...

```
public class Sample
{
    public static void main(String[] args)
    {
        GameArena arena = new GameArena(500, 300);
        Ball b = new Ball(250, 150, 20, "GREEN");

        arena.addBall(b);

        while(true)
        {
            arena.pause();
        }
    }
}
```

Getting Creative

Imagine you want to build a simple computer game based on the GameArena classes. You plan to stick together multiple shapes to create objects and characters in your game.

Do some brainstorming in your group to generate an idea for a simple game. You're free to imagine any game you want but it should have:

- A Goodie (the player, e.g. a person, a dog, a car, a big honking space invaders gun, etc.)
- Baddies (things that player interacts with, e.g. cats, alien spaceships, road hazards, etc).

For each type of character in your game (e.g. a Goodie or Baddie), assign each team member the task of the following:

- Create a Java class to represent that character in your game. Choose an appropriate name for the class.
- Add private instance variables to represent the `x` and `y` location of your character.
- Add private instance variables in your class to hold the rectangles that make up the character. Using an array here might be a good idea if you want to keep your code concise – especially if you plan to have a lot of Rectangles!
- Write a constructor that takes an `x` and `y` location as its parameters. Use this constructor to initialise your location instance variables and to create your rectangles. Get creative here to make a shape that looks the part!
- Write a public method called `addTo`. This method should take a `GameArena` object as its sole parameter. When called, the method should add all the rectangles in your object to the GameArena provided.
- Write a public method called `move`. This method should move that character in the x and y direction relative to its current position. This method should update the instance variables representing the location of your character and in each of the Rectangle instances used to draw the character.

Write a main method:

- Create a separate class named after your game, and write a `main` method in it.
- Create instances of `GameArena` and the classes you just wrote to represent the characters in your game.

- Using the methods you have just written, add your characters to your GameArena.
- Write a loop that calls the move method on your Goodie. Your Goodie should move based on the up, down, left and right cursor keys. *HINT: If you're wondering how to get user input? Checkout the JavaDoc documentation for the GameArena class for an easy way to detect these key presses…*
- Update your loop so your baddies move according to some simple rules…

Gasp in awe at your wonderful creation!

Note in particular how grouping the data and code related to a particular character into its own class has helped to keep your code simple as the complexity of your program grows…

Feel free to build upon your work to include other shapes, characters and user input as you wish.

Next week we will continue to build on this work… So get creative with your teammates, explore Java and OO programming, and show me what you can do!

SCC.121: ALGORITHMIC EFFICIENCY

This week's lab exercises will focus on the time complexity of algorithms in terms of the big-O, big-Ω and big-θ notations. Big-O defines an asymptotic upper bound, big-Ω an asymptotic lower bound and big-θ a tight bound on an algorithm efficiency. These are standard metrics for defining algorithm efficiency and enable us to compare the efficiency of different algorithms using an accepted standard according to how their run-time (or space requirements) grow as the input size grows. This week's lab exercises can all be done using *pen and paper*.

TASKS

1. For each of the following pairs of functions, state which one grows faster asymptotically for input of size n, in other words, is worse in terms of efficiency.

   - $f(n) = n^{1000} + 53$ and $g(n) = log_{10}(n) + 1.2^n$
   - $f(n) = n^{169} log_{10}(n)$ and $g(n) = n^{170} + 6n$
   - $f(n) = 6000n^{0.0001} + 19$ and $g(n) = log_{10}(n) + 1$

2. Prove that $T(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3$ is Big-$O(n^3)$ using the formal definition of the Big-O notation.

3. Prove that $T(n) = 3n^2 + 5n + 1$ is Big-$\Theta(n^2)$ using the formal definition of the Big-Θ notation.

4. What is the worst-case complexity (in Big-Θ) of the following code fragment? Assume the inputs are two arrays of size N and M respectively. In the code, "condition" takes constant time, "sequence1" takes $\Theta(N)$ time, and "sequence2" takes $\Theta(N^2)$ time. What about if the "condition" takes $\Theta(N)$ time instead?

```
for (int i=0; i<M; i++){
    if (condition){
        sequence1
    }
    else{
```

```
        sequence2
    }
}
```

5. The following code fragment takes $\Theta(N^2)$ time in the worst case. Make one change (i.e. only add or replace a single character) to the code fragment to get each of the complexities listed below (there can be more than one way to get some of these)

   - $\Theta(N^4)$
   - $\Theta(N^3)$
   - $\Theta(N^2)$, that is, make a change to get the same time complexity
   - $\Theta(N)$
   - $\Theta(1)$

```
for (i = 0; i < N; i++){
    for (j = 0; j < N; j++){
        ... some Θ(1) code ...
    }
}
```

6. What is the best-case and worst-case complexity (in big-O, big-Ω and big-Θ) of the standard stack operations (e.g. push, pop)?

7. Say we are given the names of n cities in an array and a function "dist" that returns distance given two cities given their names in constant time, that is, $\Theta(1)$ in the worst case. Also assume string operations are $\Theta(1)$. State the worst-case complexity (in Big-Θ) of the following algorithms?

   - Print all city names
   - Find city with longest name
   - Find distance between two cities given their names
   - Find city with second longest name
   - Find distances between all pairs of cities

8. Timsort is a sorting algorithm that has best case time complexity $\Theta(n)$ and worst case time complexity (n log n). Based on the formal definitions of Big-Θ, Big-O and Big-Ω are the following statements true or false?

   - Timsort is $\Omega(nlogn)$
   - Timsort is $O(nlogn)$
   - Timsort is $O((logn)^2)$ in the worst case

SCC.131: ASSEMBLY DEVELOPMENT TASKS

GOALS

In this session, we aim to enhance our understanding and proficiency in Assembly language programming. By engaging in practical exercises, participants will revisit essential programming constructs and instruction sets. By the end of this lab, you should be able to:

- ☐ Write functions in Assembly language.
- ☐ Utilize loops and conditional statements effectively.
- ☐ Perform read/write operations on memory.

THE TEMPLATE

To implement your code, we provide this week a template project, similar to the code we provided in week 14. You should edit the code in `src/main.S` file. This file is pre-populated with several functions and an integer array named `arr`, which can hold 10 integer values. The project code is designed for debugging with the micro:bit platform. You can also use the cpulator platform to implement the main.S functionality. The provided source code contains five empty functions:

- `_start` : Serves as the program's entry point, containing calls to various functions followed by an infinite loop to keep the CPU active.
- `arr_sum` : A placeholder for your implementation of a function that calculates the sum of the array's elements.
- `arr_mean` : Another placeholder where you should calculate the average of the values stored in `arr`.
- `arr_sort` : Here, you're tasked with writing a function that sorts the elements of `arr`.

```
.syntax unified


.data
  arr: .word 100, 20, 50, 40, 30, 80, 70, 60, 90, 10
```

```
    .text

    .global arr_sum
arr_sum:
  @ Add code to compute sum
  bx lr

    .global arr_mean
arr_mean:
  @ Add code to estimate mean
  bx lr

    .global arr_sort
arr_sort:
  @ TODO add code to sort data
  bx lr

  .global _start
_start:

  bl arr_sum

  bl arr_mean

  bl arr_sort

loop:
   nop
  b loop
```

Currently, only the `_start` function is implemented. Your task is to implement basic arithmetic

operations as specified below.

TASK 1: SUMMING RANDOM NUMBERS

Implement the code in `arr_add`. This function should iterate over `arr`, summing its elements, and return the total sum.

TASK 2: MEAN ESTIMATION

Your mean esitmator code should be included in the `arr_mean` function. The function should compute the mean of the array's elements. You can estimate the mean using the formula below for successive elements, starting with the first:

$$mean_n = \begin{cases} val_1, & \text{if } i = 1 \\ \frac{mean_{n-1} + val_n}{2}, & \text{if } i > 1 \end{cases}$$

Here's a C-style pseudocode example for clarity:

```
int mean(int arr[], int n) {
    int mean = arr[0];
    for (int i = 1; i < n; i++) {
        mean = (mean + arr[i]) / 2;
    }
    return mean;
}
```

TASK 3: SORTING NUMBERS

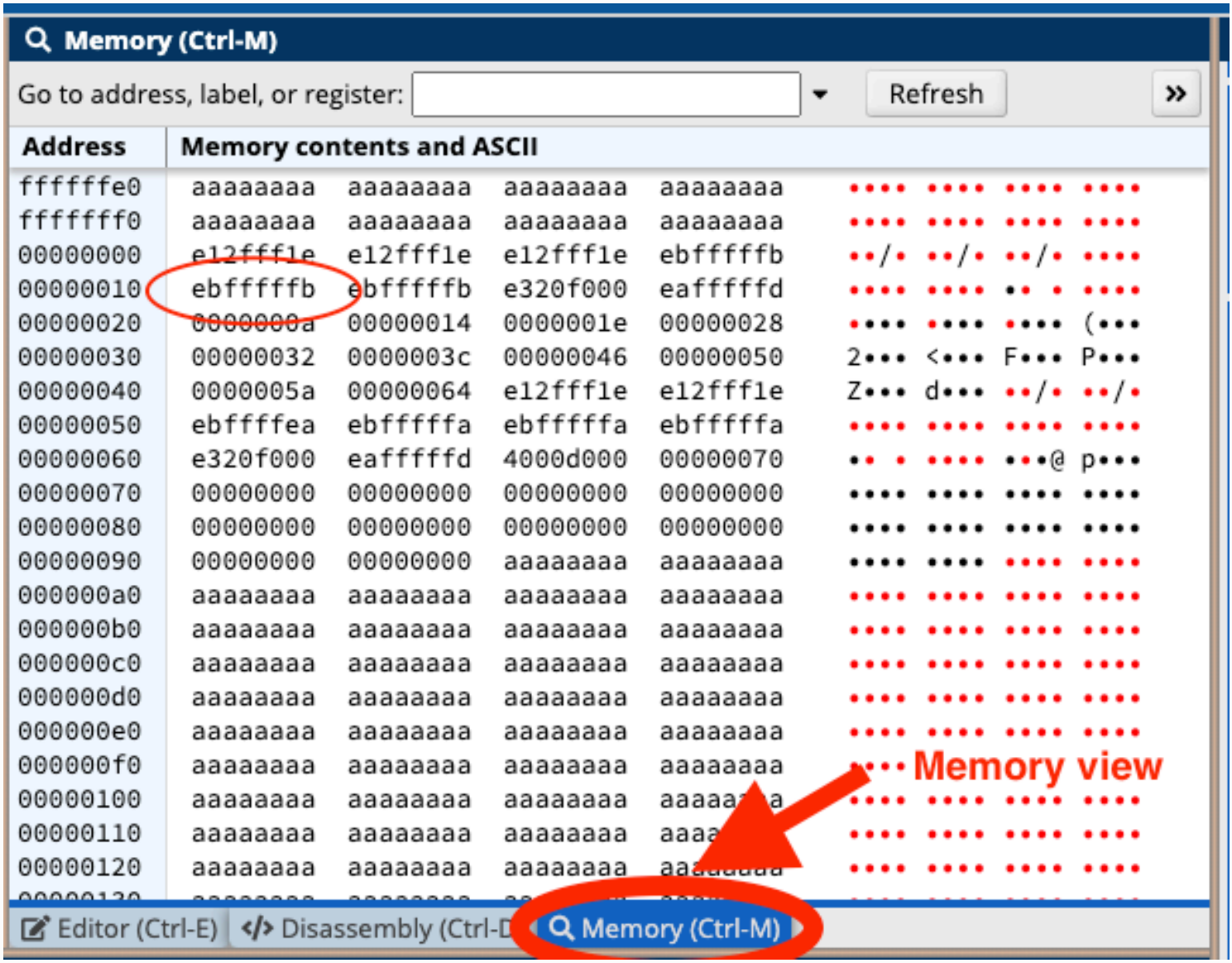
The Bubble Sort algorithm is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The list sorting step is repeated until the list is sorted. The algorithm gets its name because smaller elements "bubble" to the top of the list (beginning of the list) with each iteration. Here's a basic pseudo algorithm for Bubble Sort:

```
void bubbleSort(int arr[], int size) {
    int swapped = 1;
    while(swapped)
      swapped = 0;
      for (int j = 0; j < size - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {
          // Swap arr[j] and arr[j + 1]
          int temp = arr[j];
          arr[j] = arr[j + 1];
          arr[j + 1] = temp;
          swapped = 1;  // Mark that a swap occurred
        }
      }
    }
}
```

Implement the Bubble Sort algorithm in `arr_sort` to order the elements within `arr`. Bubble Sort is a straightforward algorithm that repeatedly traverses the list, comparing and swapping adjacent elements as needed until the list is sorted. Although simple, it is not the most efficient for large datasets due to its O(n^2) time complexity.

ADDITIONAL NOTES

In order to inspect the content of the array, you have to inspect the memory content. In the cpulator platform you can use the memory view pane in the window.



For the micro:bit debugger, use the memory view feature in VS Code, found at the bottom of the window. By navigating to address $0x20000000$, you'll reach the start of the `arr` memory region. Here, you can inspect the array's current state by pausing program execution.

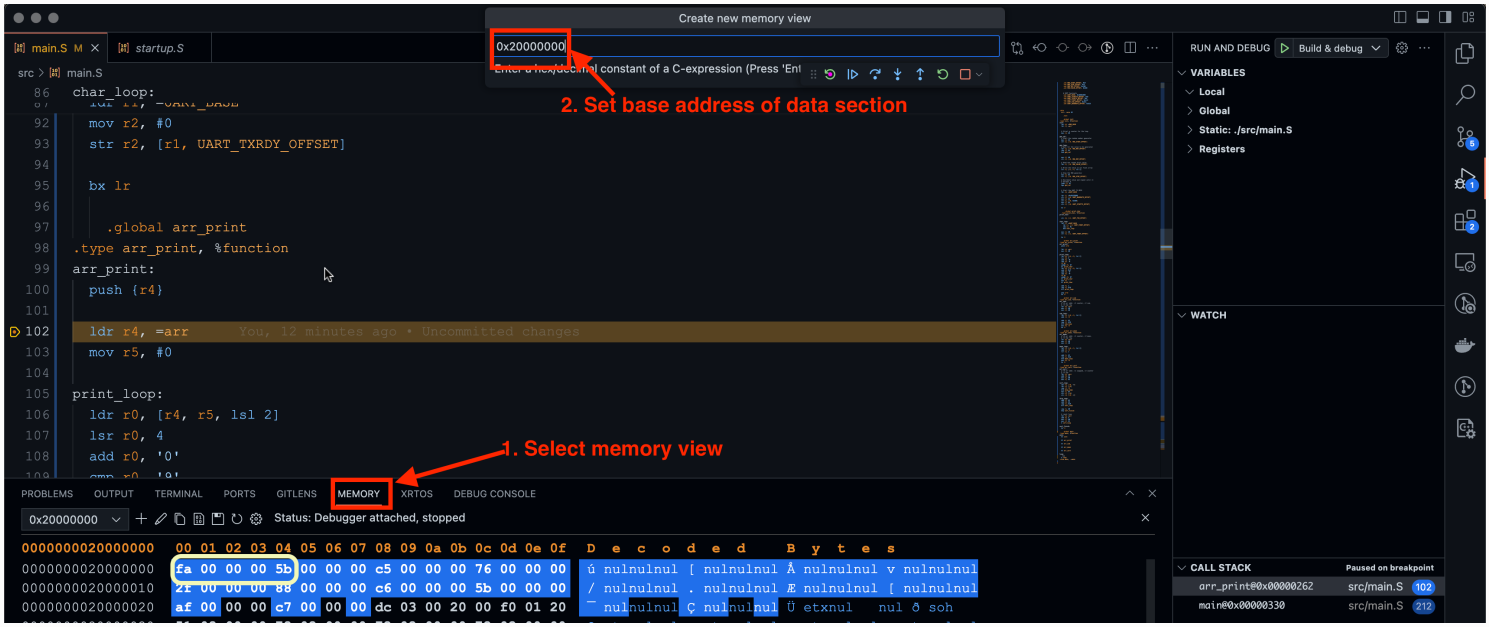


Figure 1: Sample memory view snapshot.

HACKER EDITION

SCC.121: ALGORITHMIC EFFICIENCY

1. Prove that $T(n) = log_2(n)$ is $O(log_b(n))$.

SCC.131: BINARY SEARCH

Binary search is a simple algorithm to search for a value in a sorted array. It works by repeatedly dividing the search interval in half. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

```C
int binary_search(int arr[], int len, int target, int start, int end) {
    while (start <= end) {
        int mid = start + (end - start) / 2; // Prevent potential overflow
        if (arr[mid] == target) {
            return mid;
        }
        else if (arr[mid] < target) {
            start = mid + 1;
        }
        else {
            end = mid - 1;
        }
    }
    return -1; // Target not found
}
```

Example

Consider the sorted array [1, 3, 5, 7, 9, 11, 13] and target value 7:

1. The middle element is 7, which is the target value.
2. Return the index of 7, which is 3.

Implement in ARM assembly the code that implement binary search, once you completed the implementation of the `arr_sort` function.