CP40061E – A1

PROGRAMMING Portfolio

Table of Contents

[1. Task 1: Output 1](#_Toc181698317)

[1.1. Analysis of problem 1](#_Toc181698318)

[1.2. Flowchart 1](#_Toc181698319)

[1.3. Pseudocode 1](#_Toc181698320)

[1.4. Java code 1](#_Toc181698321)

[1.5. Evaluation 1](#_Toc181698322)

[2. Task 2: Input, output and processing 2](#_Toc181698323)

[2.1. Analysis of problem 2](#_Toc181698324)

[2.2. Flowchart 2](#_Toc181698325)

[2.3. Pseudocode 2](#_Toc181698326)

[2.4. Java code 2](#_Toc181698327)

[2.5. Evaluation 2](#_Toc181698328)

[3. Task 3: Branching and loops 3](#_Toc181698329)

[3.1. Analysis of problem 3](#_Toc181698330)

[3.2. Flowchart 3](#_Toc181698331)

[3.3. Pseudocode 3](#_Toc181698332)

[3.4. Java code 3](#_Toc181698333)

[3.5. Evaluation 3](#_Toc181698334)

[4. Task 4: Arrays 4](#_Toc181698335)

[4.1. Analysis of problem 4](#_Toc181698336)

[4.2. Flowchart 4](#_Toc181698337)

[4.3. Pseudocode 4](#_Toc181698338)

[4.4. Java code 5](#_Toc181698339)

[4.5. Evaluation 5](#_Toc181698340)

[5. Task 5: Objects and classes 6](#_Toc181698341)

[5.1. Analysis of problem 6](#_Toc181698342)

[5.2. Flowchart 6](#_Toc181698343)

[5.3. Pseudocode 6](#_Toc181698344)

[5.4. Java code 7](#_Toc181698345)

[5.5. Evaluation 7](#_Toc181698346)

# Task 1: Output

A screenshot of a computer program

Description automatically generated

## Analysis of problem

*How are you going to solve the problem?*

First, when I identified the problem, I used variables that we learned on first week, to determine and store each repeating line pattern, StarLine1, StarLine2, and Equaline. After determining the variables, I used the for loop to print the lines of code for the stars and stripes. After that, I used an if-else conditional structure inside the for loop that allows me to print alternating between the two StarLines, since one of them is even and the other odd. I used the modulo operator % that allowed me to alternate when one is odd and when the other is even. Because the flag pattern is stars and lines, I had to make a separation so that at the time of output the flag would be printed correctly. I used the techniques I learned in lessons 1 and 2 to solve the flag problem effectively and simply, so that my code would be easy to read and easy to modify, with a simple and uncluttered structure.

*What programming constructs and techniques are you using? Why?*

All techniques used were learned in the classroom, facilitating the readability of the code by keeping it clean and readable, ensuring flexibility by allowing you to adjust or change variables or loop patterns, efficiently reusing patterns and loops to reduce the amount of code and potential errors, and separating the codes into distinct parts since the patterns are not the same for star lines and equal lines.

Variables - because it makes the code less cluttered and easier to modify.

For Loop - because it allows me to repeat patterns in a concise and scalable way, reducing code duplication.

Conditional If-Else - because it ensures that my code is switched correctly while keeping it clear and easy to follow.

Module Operator - because it is the most efficient way to switch between loop-based patterns, it simplifies the logic and keeps the algorithm clear and easy to read.

Logical Separation - because it reflects the output of the flag correctly and keeps my code more organized and easier to modify and debug.

## Flowchart

Uma imagem com diagrama, Esquema, Desenho técnico, file

Descrição gerada automaticamente

## Pseudocode

Start

Declare StarLine1 as string with value "\* \* \* \* \* \* ==================================\n"

Declare StarLine2 as string with value " \* \* \* \* \* ==================================\n"

Declare EqualLine as string with value "=========================================\n"

For i from 0 to 8 do (total of 9 lines)

If i is even then

Print StarLine1

Else

Print StarLine2

End If

End For

For i from 0 to 8 do (total of 9 lines)

Print EqualLine

End For

End program

## Java code

public class AmericanFlagTest { /\**The class name of my program\**/

public static void main(String[] args) {

String StarLine1 = "\* \* \* \* \* \* ==================================\n"; /\*Used to declare variable\*/

String StarLine2 = " \* \* \* \* \* ===================================\n"; /\*Used to declare variable\*/

String EqualLine = "==============================================\n"; /\*Used to declare variable\*/

for (int i = 0; i < 9; i++) { /*Start the for loop*/

/\**for loop to differentiate between even and odd star lines\**/

if (i % 2 == 0) { System.out.print(StarLine1); }

else { System.out.print(StarLine2); } }

/\**Used to loop the equal lines\*/*

for (int i = 0; i < 9; i++) {

System.out.print(EqualLine);

}

}

}

## Evaluation

*What are the strengths of your solution?*

*The strengths of my code are the simplicity written in a simple and direct manner, easy to understand and modify. The clean structure is easy to read and the correct use of variables and loops gives a good structure to the code.*

*What are the weaknesses?*

*The weak points are that I could encapsulate the logic in functions or methods but in classes 1 and 2 I still hadn't learned that, and the way the flag is standardized doesn't allow flexibility in customization.*

*In what way could it be improved?*

For better optimization, I would make available to the user a way and tools to adjust the flag, as well as its customization, allowing the creation of other types of flags, and more clarity in the comments for better future modification if necessary.

# Task 2: Input, output and processing

A white background with black text

Description automatically generated

## Analysis of problem

*How are you going to solve the problem?*

*The problem is to create a program that calculates the BMI and categorizes it. So I will create an input that takes the users' information about weight, height and gender. I will calculate using the formula BMI = weight / (height \* height), from which it will be categorized based on the values ​​obtained in Underweight: BMI < 18.5, Healthy Weight: 18.5 <= BMI < 25, Overweight: 25 <= BMI < 30 any value above is Obese. This will determine the BMI of each user .*

*What programming constructs and techniques are you using? Why?*

*My code combined the use of different techniques, such as variables, loops, user input/outputs that were well chosen to accurately determine the BMI, obtaining a good user experience, efficiency in using the program and good readability in the code.*

*I started by using the java.util.scanner class that allows the user to enter information about weight, height and gender, I also used system.out.print and println for outputs. Then I determined the variables in double that allows the insertion of decimal numbers, I used char to determine the gender in M ​​or F as it only allows storing a single character. I used boolean as continueCalculation to control the main loop and allow the calculation of the BMI, perfect condition to determine in true or false and make the loop continue efficiently and finally the string response that allows the user to answer yes or no about calculating the BMI again. As previously mentioned, I used loops to construct the code and calculate the BMI. The while loop allowed multiple calculations while it was true. The try-catch loop was used in case the user entered something different that would result in an error, preventing the program from crashing or having a bug. The if-else statement was used to classify the BMI, which had multiple exclusive conditions. I also used some arithmetic operators such as /,\*, comparison operators <,>=, ==. Logical operator &&, which were essential in the calculations and in the data manipulation.*

## Flowchart

Uma imagem com texto, diagrama, número, file

Descrição gerada automaticamente

## Pseudocode

*Start BMICalculator*

*Import the class java util scanner*

Declare *weight as double* with *real number* value

Declare *height as double* with *real number* value

Declare *bmi as double* with *real number* value

Declare *gender as character* with *text* value

*Declare continueCalculation as Boolean with true or false value*

*continueCalculation = true*

*Scanner scanner = new Scanner (System.in)*

*While continueCalculation IS TRUE DO*

*Try*

*weight = scanner.nextDouble()*

*height = scanner.nextDouble()*

*gender = scanner.next().charAt(0)*

*bmi = weight / (height \* height) // Calculate BMI*

*System.out.println("Your BMI is: " + bmi)*

*If gender IS 'M' OR gender IS 'F' THEN*

*If bmi < 18.5 THEN*

*System.out.println("You are underweight.")*

*Else If bmi >= 18.5 AND bmi < 24.9 THEN*

*System.out.println("You have a normal weight.")*

*Else If bmi >= 25 AND bmi < 29.9 THEN*

*System.out.println("You are overweight.")*

*Else*

*System.out.println("You are obese.")*

*End If*

*Else*

*System.out.println("Invalid gender! Please enter 'M' for male or 'F' for female.")*

*End If*

*Catch Exception*

*System.out.println("Invalid input! Please enter numeric values for weight and height.")*

*scanner.nextLine()*

*Continue*

*End Try*

*response = scanner.next()*

*continueCalculation = response.equalsIgnoreCase("yes")*

*End While*

*scanner.close()*

*System.out.println("Thank you for using the BMI calculator!")*

*End*

## Java code

*import java.util.Scanner; /\*Scanner is used to take input from the user.\*/*  
  
 *public class BMICalculator {*  
 *public static void main(String[] args) {*  
  
 *Scanner scanner = new Scanner(System.in); /\*Scanner object is created to capture input from the user.\*/*  
  
 */\*Declare the variables\*/*  
 *boolean continueCalculation = true;/\* Variable to allow the program keep running\*/*  
 *double weight = 0;/\*Variable to store user’s weight\*/*  
 *double height = 0; /\*Variable to store user’s height\*/*  
 *double bmi = 0; /\*Variable to store user’s bmi\*/*  
 *char gender = ' '; /\*Variable to store user’s gender\*/*  
  
 */\* Loop to BMI calculations\*/*  
 *while (continueCalculation) {*  
 */\* loop allows the user to calculate BMI multiple times without restarting the program. The loop allow the program continue running as continueCalculation is true.\*/*  
 *// Input weight, height, and gender*  
 *try {*  
 *System.out.print("Input your weight in kg: ");*  
 *weight = scanner.nextDouble(); //User weight input*  
  
 *System.out.print("Input your height in meters: ");*  
 *height = scanner.nextDouble(); //User height input*  
  
 *System.out.print("Input your gender (M/F): ");*  
 *gender = scanner.next().charAt(0); //User gender input*  
  
 *if (weight <= 0 || height <= 0) {*  
 *System.out.println("Invalid input: Weight and height must be positive values.");*  
 *continue; //continue the loop without exiting*  
 *}*  
  
 */\*BMI calculation\*/*  
 *bmi = weight / (height \* height);*  
 *System.out.println("Your BMI is: " + bmi);*  
  
 *//Feedback to the BMI and gender*  
 *if (gender == 'M' || gender == 'F') { //Gender validation*  
 *//Classification based on BMI range*  
 *if (bmi < 18.5) {*  
 *System.out.println("You are underweight.");*  
 *} else if (bmi >= 18.5 && bmi < 24.9) {*  
 *System.out.println("You have a normal weight.");*  
 *} else if (bmi >= 25 && bmi < 29.9) {*  
 *System.out.println("You are overweight.");*  
 *} else {*  
 *System.out.println("You are obese.");*  
 *}*  
 *} else {*  
 *// Error message to invalid input*  
 *System.out.println("Invalid gender! Please enter 'M' for male or 'F' for female.");*  
 *}*  
  
 *} catch (Exception e) {*  
 *//Handle invalid numeric input for weight or height*  
 *System.out.println("Invalid input! Please enter numeric values for weight and height.");*  
 *scanner.nextLine(); //Clear the invalid input from the buffer*  
 *continue; //continue the loop without exiting*  
 *}*  
  
 *// Ask the user if they wanna calculate BMI again*  
 *System.out.print("Do you want to calculate again? (yes/no): ");*  
 *String response = scanner.next(); //user response*  
 *continueCalculation = response.equalsIgnoreCase("yes");*  
 *}*  
 *// Close the scanner and thank the user*  
 *scanner.close();*  
 *System.out.println("Thank you for using the BMI calculator!");*  
 *}*  
 *}*

## Evaluation

*What are the strengths of your solution?*

*Some strong points are clear comments that explain the purpose of different variables, functions, etc. and make the program easier to understand. Also, user input validation that does not allow negative values, that preventing input errors. Loops that allow the user to calculate their BMI multiple times without having to restart the program.*

*What are the weaknesses?*

*The weak points are that the program is all done in main and this makes the code less organized, makes the program difficult to read and makes it more difficult to extend or maintain.*

*In what way could it be improved?*

*To improve the program, I can increase the user input validation to check for non-positive values and add a more realistic range for the user's height and weight. I can also consider a unit conversion to accept both meters and centimetres, too improving the user experience and reducing the chances of errors.*

*D*

# Task 3: Branching and loops

A close up of text

Description automatically generated

## Analysis of problem

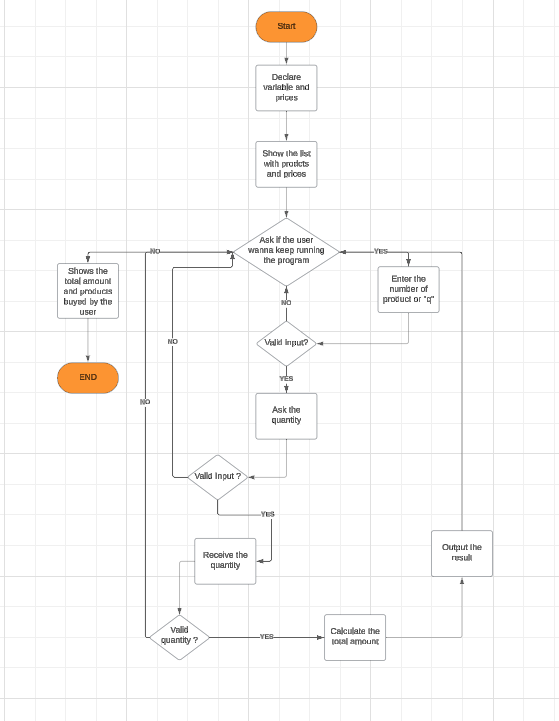
*How are you going to solve the problem?*

*First, as soon as I understood the problem, I declared variables to set the products and prices, then I took the information entered by the user and validated it so that no errors would occur. With the user's information, I developed a code for the sum and final result of the total costs and used loops for the user to decide when he would like to finish the purchase, thus finalizing the order with the total sum of the products.*

*What programming constructs and techniques are you using? Why?*

*I used variables to store the names and prices of the products, then I used loops to allow the user to continue selecting products and quantities as much as they want, then I used if-else conditionals to validate the user's inputs so that no errors occur or negative numbers are entered, I also used mathematical operations to multiply, add the price and quantity of the products, and I used outputs to show error messages, to show the purchase menu, prices and quantities and the final price of the products.*

## Flowchart



## Pseudocode

*Start*

*Set product1 = "Monitor", price1 = 249.0*

*Set product2 = "CPU", price2 = 599.0*

*Set product3 = "GPU", price3 = 1950.0*

*Set product4 = "RAM", price4 = 78.0*

*Set total = 0*

*Print "Welcome to the Online Store!"*

*Print "Here are the available products:"*

*Print "1. Monitor - 249.0£"*

*Print "2. CPU - 599.0£"*

*Print "3. GPU - 1950.0£"*

*Print "4. RAM - 78.0£"*

*While true do*

*Print "Enter the product number to buy (or 'q' to quit): "*

*Input userInput*

*If userInput Equal "q" Or userInput Equals "Q" Then*

*BREAK*

*End if*

*Try*

*Set productNumber = PARSE userInput AS INTEGER*

*If productNumber < 1 Or productNumber > 4 Then*

*Print "Invalid product number. Please try again."*

*CONTINUE*

*End if*

*Print "Enter the quantity: "*

*Input quantity*

*If quantity < 0 THEN*

*Print "Quantity cannot be negative. Please try again."*

*CONTINUE*

*End if*

*Set cost = 0*

*If productNumber Equals 1 Then*

*cost = price1 \* quantity*

*Print "Added " + quantity + " Monitor(s) to your cart."*

*Else if productNumber Equals 2 Then*

*cost = price2 \* quantity*

*Print "Added " + quantity + " CPU(s) to your cart."*

*Else if productNumber Equals 3 Then*

*cost = price3 \* quantity*

*Print "Added " + quantity + " GPU(s) to your cart."*

*Else if productNumber Equals 4 Then*

*cost = price4 \* quantity*

*Print "Added " + quantity + " RAM(s) to your cart."*

*End if*

*total = total + cost*

*Print "Current total: " + total + “£”*

*Catch NumberFormatException*

*Print "Invalid input. Please enter a number or 'q' to quit."*

*End try*

*End while*

*/\*Display final total and exit message\*/*

*Print "Thank you for shopping with us!"*

*Print "Your total amount is: " + total + ” £”*

*END*

## Java code

import java.util.Scanner; /\**Scanner is used to take input from the user.\**/

public class OnlineStore { /\**Name of the class\**/

public static void main(String[] args) { /\**Variables for product names and prices\**/

String product1 = "Monitor"; /\*Variable to name product1\*/  
double price1 = 249.0; /\*Variable to price product1\*/   
  
String product2 = "CPU"; /\*Variable to name product2\*/   
double price2 = 599.0; /\*Variable to price product2\*/   
  
String product3 = "GPU"; /\*Variable to name product3\*/   
double price3 = 1950.0; /\*Variable to price product3\*/   
  
String product4 = "RAM";/\*Variable to name product4\*/   
double price4 = 78.0;/\*Variable to price product4\*/   
  
/\*Scanner for user input\*/   
Scanner scanner = new Scanner(System.in);  
double total = 0;   
  
/\*Output interface to the program\*/   
System.out.println("Welcome to the Online Store!");   
System.out.println("Here are the available products:");   
System.out.println("1. " + product1 + price1 + "£");   
System.out.println("2. " + product2 + price2 + "£");   
System.out.println("3. " + product3 + price3 + "£");   
System.out.println("4. " + product4 + price4 + "£");  
  
while (true) {   
 /\*Ask the user to select a product\*/   
 System.out.println("Enter the product number to buy (or 'q' to quit): ");   
 String input = scanner.nextLine();   
  
 /\*Check if the user wants to quit\*/   
 if (input.equalsIgnoreCase("q")) {   
 break;   
 }   
  
 try {   
 int productNumber = Integer.parseInt(input);   
  
 /\*Validation of product number\*/   
 if (productNumber < 1 || productNumber > 4) {   
 System.out.println("Invalid product number. Please try again.");   
 continue;   
 }   
  
 /\*Ask the user for the quantity\*/   
 System.out.println("Enter the quantity: ");   
 int quantity = Integer.parseInt(scanner.nextLine());   
  
 if (quantity < 0) {   
 System.out.println("Quantity cannot be negative. Please try again.");   
 continue;   
 }   
  
 /\*Add the cost of the selected product to the total\*/   
 double cost = 0;   
 if (productNumber == 1) {   
 cost = price1 \* quantity;   
 System.out.println("Added " + quantity + " " + product1 + "'s to your cart.");/\* Output price and quantity to the product one\*/   
 } else if (productNumber == 2) {   
 cost = price2 \* quantity;   
 System.out.println("Added " + quantity + " " + product2 + "'s to your cart.");   
 } else if (productNumber == 3) {   
 cost = price3 \* quantity;   
 System.out.println("Added " + quantity + " " + product3 + "'s to your cart.");   
 } else if (productNumber == 4) {   
 cost = price4 \* quantity;   
 System.out.println("Added " + quantity + " " + product4 + "'s to your cart.");   
 }   
  
 total += cost;   
 System.out.println("Current total: " + total + "£");   
  
 } catch (NumberFormatException e) {   
 System.out.println("Invalid input. Please enter a number or 'q' to quit.");   
 }   
}   
  
/\*Total amount output\*/   
System.out.println("Thank you for shopping with us!");   
System.out.println("Your total amount is: " + total + "£");   
  
scanner.close();

}

}

## Evaluation

*What are the strengths of your solution?*

*The strengths of my code are the simplicity with which the code was made, making it easy to read, with direct and instructive comments, well-structured with variables, loops, error validation, flexibility for the user to choose, input and outputs, the calculation of the total cost of the products well listed after all the purchase entries.*

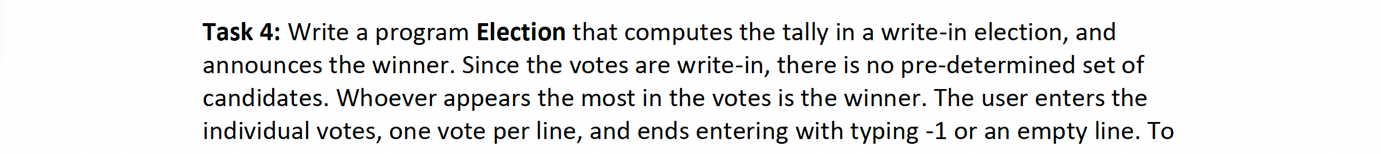
*What are the weaknesses?*

*The weak points of my program start with scalability as I had not yet learned about arrays, this limited the program to individual variables for each product, making the program more extended, the structure is also not flexible making the program only have 4 products, there is also a problem that the program does not allow you to remove items from the cart, and the program depends on the user inputting something*

*In what way could it be improved?*

*To improve, I could structure the code better and simplify it using array lists, a more diverse list of products and a cart that could delete items and change the quantity of items as well.*

# Task 4: Arrays



A screenshot of a computer program

Description automatically generated

## Analysis of problem

*How are you going to solve the problem?*

*First, when I identified the problem, I defined two arraylists, one for the candidates and the other for the votes. I defined a boolean variable for the loop to continue running the program every time it is true or exit when the user wants. In the loop, the code stores the candidate's name and the number of votes, adding them until the count is finished, or creating a new entry every time a new candidate is inserted. The result is looped through two lists that show the candidates and another that shows the number of votes, to finally show the winner and how many votes he had.*

*What programming constructs and techniques are you using? Why?*

*I chose to use the arraylist instead of the array because it allows me to store the candidates in an optimized and practical way, making my code cleaner and more efficient. I used a while loop to continue the code even when a candidate receives a vote or until the user wants to exit. I used the for loop to calculate the votes for each candidate and show the results. I also used the if.else to check the inputs used by the user, verify if each candidate exists and determine the number of votes for each one. I used the input scanner for the user to enter information. I used the try-catch to validate inputs and avoid errors or wrong inputs. I also used a string trim manipulation that removes the spaces around the inputs, helping to keep the format in which the candidates appear consistent. I also used a logical comparison to compare the votes and see who was the winner. Finally, the displayResultsAndWinner encapsulation was a logic used to determine the winner.*

## Flowchart

*Paste a copy of a flowchart for your solution here.*

## Pseudocode

*Start*

*Initialize an empty list `candidates` to store candidate names*

*Initialize an empty list `counts` to store the corresponding vote counts for each candidate*

*Set a boolean variable `looping` to `true` to control the input loop*

*Create a Scanner object `input` to read user inputs*

*Print the election header message:*

*Enter votes type -1 or press Enter to finish*

*WHIle looping is true*

*Try*

*Print Enter the candidate name*

*Read user input as a string `candidate` and trim leading/trailing spaces.*

*If `candidate` is equal to "-1" OR `candidate` is empty:*

*Set `looping` to `false` to exit the loop.*

*Break from the loop.*

*Check if `candidate` exists in the `candidates` list:*

*If `candidate` exists:*

*Find its index in the `candidates` list.*

*Increment the corresponding vote count in the `counts` list at the same index.*

*Else:*

*Add `candidate` to the `candidates` list.*

*Add `1` to the `counts` list as the initial vote count.*

*Catch any input exceptions:*

*Print "Invalid input! Please try again."*

*Clear the scanner buffer.*

*Call the `displayResultsAndWinner(candidates, counts)` method to display results.*

*Method displayResultsAndWinner(candidates, counts)*

*If `candidates` list is empty*

*Print No votes.*

*Return from the method*

*Print And the results for the Election is*

*Initialize variables*

*winner to store the name of the candidate with the most votes*

*maxCounts to store the maximum vote count initially `0`*

*For each index `i` in the range of the size of the `candidates` list*

*Print the candidate name at index `i` and their vote count from the `counts` list.*

*If the vote count at index `i` is greater than `maxCounts`*

*Update `maxCounts` with the vote count at index `i`.*

*Update `winner` with the candidate name at index `i`*

*Print And the winner with most votes is winner with maxCounts votes*

## Java code

import java.util.ArrayList;

import java.util.Scanner;

public class Election {/\*Class name\*/

public static void main(String[] args) {  
 /\*create the list for candidates and vote counts\*/  
 ArrayList<String> candidates = new ArrayList<>();/\*arraylist to candidates\*/  
 ArrayList<Integer> counts = new ArrayList<>();/\*arraylist to counts\*/  
 boolean looping = true;  
   
 Scanner input = new Scanner(System.in);/\*Scanner to user inputs\*/  
   
 System.out.println("##############################################\n" +  
 "Enter votes (type -1 or press Enter to finish): \n" +  
 "##############################################\n");/\*print out the main head\*/  
 /\*loop to designate the candidate and votes\*/  
 while (looping) {  
 try {  
 System.out.println("Enter the candidate name: ");  
 /\*input the name of candidate\*/  
 String candidate = input.nextLine().trim();  
 /\*stop the program if the user types "-1" or leaves input empty\*/  
 if (candidate.equals("-1") || candidate.isEmpty()) {  
 looping = false;  
 break;}  
 /\*check if the candidate exists\*/  
 int index = candidates.indexOf(candidate);  
 if (index != -1) {  
 /\*add vote if candidate exists\*/  
 counts.set(index, counts.get(index) + 1);  
 } else {  
 /\*add new candidate to list\*/  
 candidates.add(candidate);  
 counts.add(1);  
 }  
 /\* validate the loop and restart the input\*/  
 } catch (Exception e) {  
 System.out.println("Invalid input! Please try again.");  
 input.nextLine();  
 }  
 }  
 /\*shows the results\*/  
 displayResultsAndWinner(candidates, counts);}  
/\*logic to results and look the winner\*/  
private static void displayResultsAndWinner(ArrayList<String> candidates, ArrayList<Integer> counts) {  
 if (candidates.isEmpty()) {  
 System.out.println("No votes.");  
 return;}  
 System.out.println("\n And the results for the Election is: ");  
 /\*variable to track winner\*/  
 String winner= null;  
 int maxCounts = 0;  
 /\*Loop to calculate the votes to each candidate\*/  
 for(int i = 0; i < candidates.size(); i++ ) {  
 System.out.println(candidates.get(i) + ": " + counts.get(i) + " votes");  
 /\*statement to get the winner\*/  
 if (counts.get(i) > maxCounts) {  
 maxCounts = counts.get(i);  
 winner = candidates.get(i);}  
 }  
 /\*Output of the winner and how many votes\*/  
 System.out.println ("And the winner with most votes is: " + winner + " with " + maxCounts + " votes!");

}

}

## Evaluation

*What are the strengths of your solution?*

*The strengths of my code are that by using arraylist I have control over the number of candidates without having to predefine it, making my program flexible without making it predictable. Also, the input vote process and the winner result were done separately, thus giving greater control over the manipulation of the code, which makes the code easier to read and change. The program also has user input validation, to make sure that no wrong entries are made. The feedback prompts are also a strong point because they are intuitive and help the user to use the program correctly. Not to mention that the solution to the problem allows and supports any adjustment in the size of the elections, given that the arraylist has great scalability.*

*What are the weaknesses?*

*The weaknesses of my code are that it is inefficient in searching for candidates, because the arraylist searches linearly for each new vote, which can cause the search to slow down as the number of votes and candidates increases. We also assume that the program chooses a winner if there is a tie in votes. The program also has poor error control. It assumes valid inputs but does not produce more complete feedback on errors, which can end up generating more input errors on the part of the user.*

*In what way could it be improved?*

*I looked for ways to improve my program. One of them would be to redo the structure of the arraylist in hashmap, putting the names of the candidates as key and counts as value. I could also increase the result of the logical calculations to detect ties, so if one or more candidates have the same number of votes I could make a choice based on other indications. I could also improve the output and comments with more robust and complete feedback, and also provide a way for the user to change or remove a vote.*

# Task 5: Objects and classes

A screenshot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

## Analysis of problem

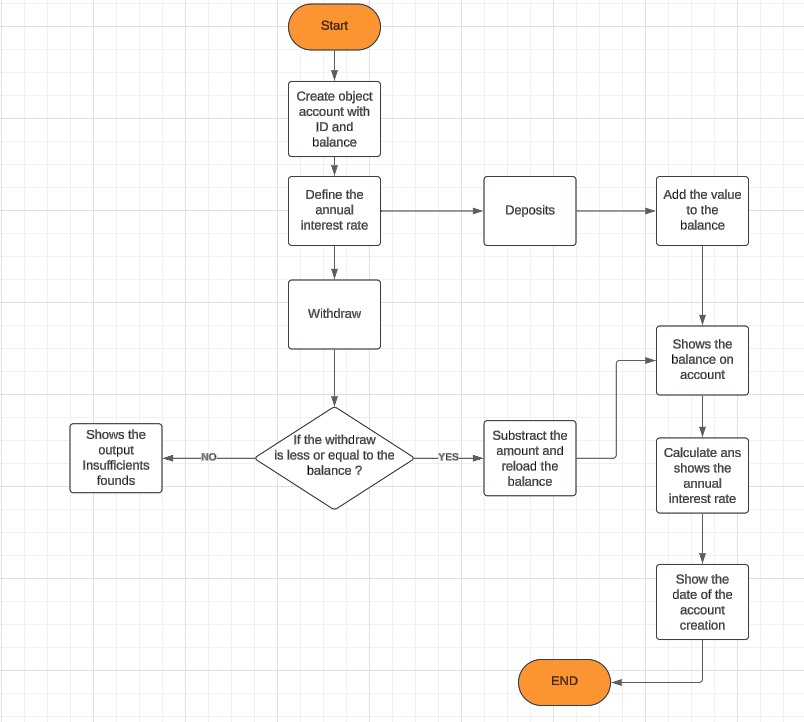
*How are you going to solve the problem?*

*When I identified the problem, I saw that it was necessary to create a Java class named account that contains attributes as defined in variables for id, balance annual interest rate and creation date. I created objects of the account class, a standard constructor and one with a specific id and balance, I provided access control to the account data fields through get and set, I implemented some methods to calculate the monthly interest, withdraw and deposit.*

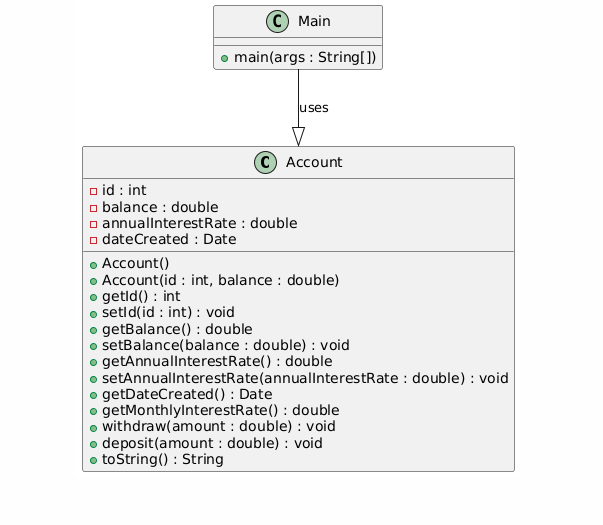
*What programming constructs and techniques are you using? Why?*

*Regarding the techniques and constructors, I used classes and objects, variables for store data with the encapsulation method that we learned last week to help protect the data from external access, constructors with different parameters to have a more flexible code when creating an account, I also used get and set for greater access control and to provide efficient ways of manipulating data, good reading and validation. I used the import java util date to show the date and time of the account creation in real time, the toString method to make it easier to print or debug and I formatted the output to show the currency value in decimal with the currency symbol.*

## Flowchart



## UML Diagram

I didn‘t understand well how to do the uml!

## Pseudocode

*Start*

*Class Account*

*private int id = 0*

*private double balance = 0*

*private double annualInterestRate = 0*

*private Date dateCreated*

*constructor Account()*

*dateCreated = currentDate*

*constructor Account(int newId, double newBalance)*

*id = newId*

*balance = newBalance*

*dateCreated = currentDate*

*function getId() returns int*

*return id*

*function getBalance() returns double*

*return balance*

*function getAnnualInterestRate() returns double*

*return annualInterestRate*

*function getDateCreated() returns Date*

*return dateCreated*

*function setId(int newId)*

*id = newId*

*function setBalance(double newBalance)*

*balance = newBalance*

*function setAnnualInterestRate(double newRate)*

*annualInterestRate = newRate*

*function getMonthlyInterestRate() returns double*

*return (annualInterestRate / 1200)*

*function withdraw(double amount)*

*if amount <= balance then*

*balance = balance - amount*

*else*

*print "Insufficient funds"*

*function deposit(double amount)*

*balance = balance + amount*

*function toString() returns string*

*return "Account{id=" + id + ", balance=" + balance + ", annualInterestRate=" + annualInterestRate + ", dateCreated=" + dateCreated + "}"*

*end class*

*program Main*

*Account myAccount = new Account(1122, 20000)*

*myAccount.setAnnualInterestRate(4.5)*

*myAccount.withdraw(2500)*

*myAccount.deposit(3000)*

*print "Balance: $" + myAccount.getBalance()*

*print "Monthly Interest: $" + (myAccount.getBalance() \* myAccount.getMonthlyInterestRate())*

*print "Date Created: " + myAccount.getDateCreated()*

*print myAccount*

*End program*

## Java code

Main.java

import java.util.Date;

public class Main {

public static void main(String[] args) {  
 Account account = new Account(1122, 20000); /\*Create an account with ID and initial balance\*/  
 account.setAnnualInterestRate(4.5); /\*et the annual interest rate\*/  
  
 account.withdraw(2500); /\*Withdraw funds\*/  
 account.deposit(3000); /\*deposit funds\*/  
  
 System.out.println("Balance: " + account.getBalance() + "£"); /\*Display current balance\*/  
 System.out.println("Monthly Interest: " + (account.getBalance() \* account.getMonthlyInterestRate()) + "£"); /\*Display monthly interest\*/  
 System.out.println("Date Created: " + account.getDateCreated()); /\* Display the account creation date\*/  
}

}

Account.java

import java.util.Date;  
  
public class Account {  
  
 private int id; /\*Unique identifier for the account\*/  
 private double balance; /\*Current balance of the account\*/  
 private double annualInterestRate; /\* Annual interest rate (as a percentage)\*/  
 private Date dateCreated; /\* The date the account was created\*/  
  
 public Account() { // Default constructor (no arguments)\*/  
 this(0, 0); /\*Calls the constructor with default values for id and balance\*/  
 }  
  
 public Account(int id, double balance) { /\*Constructor with initial values\*/ this.id = id;/\*Set the account ID\*/  
 this.balance = balance; /\*Set the initial balance\*/  
 this.annualInterestRate = 0; /\*Set the default annual interest rate\*/  
 this.dateCreated = new Date(); /\*Set the date the account was created\*/  
 }  
  
 public int getId() { /\*Get method to retrieve the account ID\*/  
 return id;  
 }  
  
 public void setId(int id) { /\*Set method to set the account ID (usually not used)\*/  
 this.id = id;  
 }  
  
 public double getBalance() { /\*Get method to retrieve the current balance\*/  
 return balance;  
 }  
  
 public void setBalance(double balance) { /\*set method to set the balance (usually not used)\*/  
 this.balance = balance;  
 }  
  
 public double getAnnualInterestRate() { /\*Get method to retrieve the annual interest rate\*/  
 return annualInterestRate;  
 }  
  
 public void setAnnualInterestRate(double annualInterestRate) { /\* Set method to set the annual interest rate\*/  
 this.annualInterestRate = annualInterestRate;  
 }  
  
 public Date getDateCreated() { /\* Get method to retrieve the account creation date\*/  
 return dateCreated;  
 }  
  
 public double getMonthlyInterestRate() { /\*Calculates and returns the monthly interest rate\*/  
 return annualInterestRate / 1200; /\*Annual rate divided by 12 months and converted to a decimal\*/  
 }  
  
 public void withdraw(double amount) { /\*Method to withdraw funds from the account\*/  
 if (amount <= balance) {  
 balance -= amount; /\*Subtract the withdrawal amount from the balance\*/  
 } else {  
 System.out.println("Insufficient funds"); /\* Inform the user about insufficient funds\*/  
 }  
 }  
  
 public void deposit(double amount) { /\*Method to deposit funds into the account\*/  
 balance += amount; /\* dd the deposit amount to the balance\*/  
 }  
  
  
 @Override

public String toString() { /\*Overrides the default to string method to a more representation\*/  
 return  
 "Account{" + "id=" + id + ", balance=" + balance + ", annualInterestRate=" + annualInterestRate + ", dateCreated=" + dateCreated + '}';  
 }  
}

## Evaluation

*What are the strengths of your solution?*

*The strong points of my code and program are that it is well commented, it meets everything that was requested such as the name class, all data fields, constructors, all necessary accessors, it is well encapsulated with private, so that there is no external access to the content, its logic is well done, the outputs are well structured and informative and the implementation of toString that allowed an easier way to print the state of the object.*

*What are the weaknesses?*

*Weak points of my code and program is that it has some problems in input validation since the withdraw and deposit do not check whether the values entered are negative or not, it also has error control limited to what was requested.*

*In what way could it be improved?*

*To improve my program I did some research and I believe I can improve input validation by adding checks for negative values, I can do more complete error control and with more robust feedback.*