***CSC 3020***

***Java Programming***

**Winter Term 2021**

**Project 01**

**50 points**

**Due 03/23/2021 (11:45 A.M.)**

**The goal of this Project is:**

1. Being able to Analyze, Design, implement, and test a practical real-world application.
2. Being able to use selections and repetition structures
3. Being able to deal with methods
4. Being able to manipulate 1D & 2D arrays.
5. Learn about user-defined classes

**Requirements:**

* In a word file:
  + Analyze each problem; outline the problem and its solution requirements. (Describe the problem including input and output in your own words.))
  + Design an algorithm to solve the problem. (Describe the major steps for solving the problem.)
    - Use UML class diagram to model the class in question 2.
* Using java IDE software, implement the algorithm.
* Test the code for each problem and verify that the algorithm works; include a screenshot of each program output.

**Restrictions:**

You must work individually. Use only material from class or from the text book (chapters 1- 9). All code must be the work of the individual. Do not share your code or copy from external resources.

**Submission**

Submit 3 .java files (1 file for Q1 and 2 files for Q2); upload all files to the Canvas by the due date. DO NOT Email your files.

**Grading:**

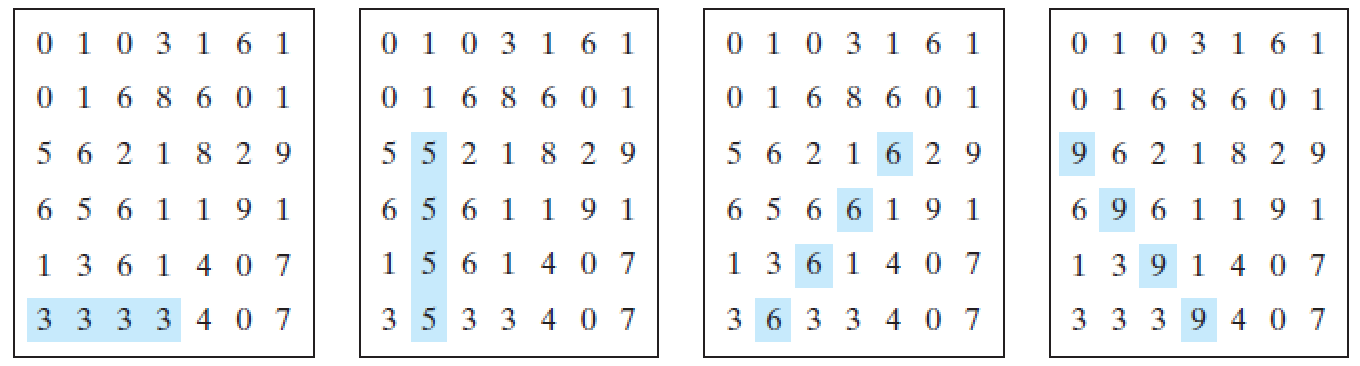
The grade of each program will be based on the creation of a program that works correctly, up to some details (50%), clear problem analysis and algorithm design (5%), the appropriate use of classes, methods and loops (25%), the production of clear output, with readable formatting and without unnecessary repetition (10%), composition of informative comments (5%), and testing the program with different inputs (5%). Programs must run.

**Question 01 - *Pattern recognition: four consecutive equal numbers* (25 points)**

Write the following method that tests whether a two-dimensional array has four consecutive numbers of the same value, either horizontally, vertically, or diagonally:

**public static boolean** isConsecutiveFour(**int**[][] values)

Write a test program that prompts the user to enter the number of rows and columns of a two-dimensional array then the values in the array, and displays true if the array contains four consecutive numbers with the same value. Otherwise, the program displays false. Here are some examples of the true cases:



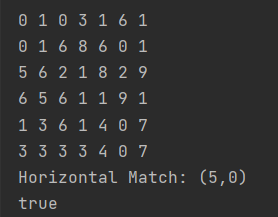
**Analysis:**

This problem requires us to design a program that can recognize 4 consecutive values in any direction along a 2D matrix. This problem also requires that the 2D Matrix be designed through user input. The user must enter the number of rows, and the number of columns for this 2D Array. Then the user must be prompted to enter values for each element in the 2D matrix. After this the program must search for four consecutive values in this 2D matrix, and then return true if there is 4 consecutive values, or false if there is not. If we loop through the array from right to left, top to bottom, then we only have to search in four directions. Right, Down, diagonal Right, and Diagonal Left, are the possible directions for us to look for four consecutive values. We will likely want the program to ouput the 2D matrix as well so that we can be sure that the values are the same as the one inputted by the user.

**Algorithm Design:**

1. Prompt user to enter number of rows and number of columns into integer values
2. create 2D array using number of rows and number of columns
3. Prompt user to enter values for each element in the 2D array.
4. then we will create a boolean variable and set it equal to the isConsecutiveFour(int[][] values) function
5. This function will then loop through each element.
6. For each element it will check if there are four consecutive values that are:
   1. horizontal
   2. vertical
   3. right diagonal
   4. left diagonal
7. When it check these functions it will first make sure that the furthest consecutive value will still be inside of the 2D array.
8. If yes, then it will check too see if each of the values in that direction are the same.
9. if yes, it will return true which will then return true in the isConsecutiveFour function.
10. this will then output the direction and the starting value of that direction for the values.
11. if no, then it will check each of those directions for the next element.
12. If it never returns true then it will output “Sorry, there is no match...” and return false.

**Screenshots:**

**A picture containing text

Description automatically generatedText

Description automatically generatedA picture containing text, electronics, keyboard

Description automatically generated**

***Question 02 - ATM machine* (25 points)**

Write a class named **Account** that contains:

* A private **int** data field named **id** for the account (default **0**).
* A private **double** data field named **balance** for the account (default **0**).
* A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0**). Assume that all accounts have the same interest rate.
* A private **Date** data field named **dateCreated** that stores the date when the account was created.
* A no-arg constructor that creates a default account.
* A constructor that creates an account with the specified id and initial balance.
* The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
* The accessor method for **dateCreated**.
* A method named **getMonthlyInterestRate()** that returns the monthly interest rate.
* A method named **getMonthlyInterest()** that returns the monthly interest not the interest rate.
* A method named **withdraw** that withdraws a specified amount from the account.
* A method named **deposit** that deposits a specified amount to the account.

Note:

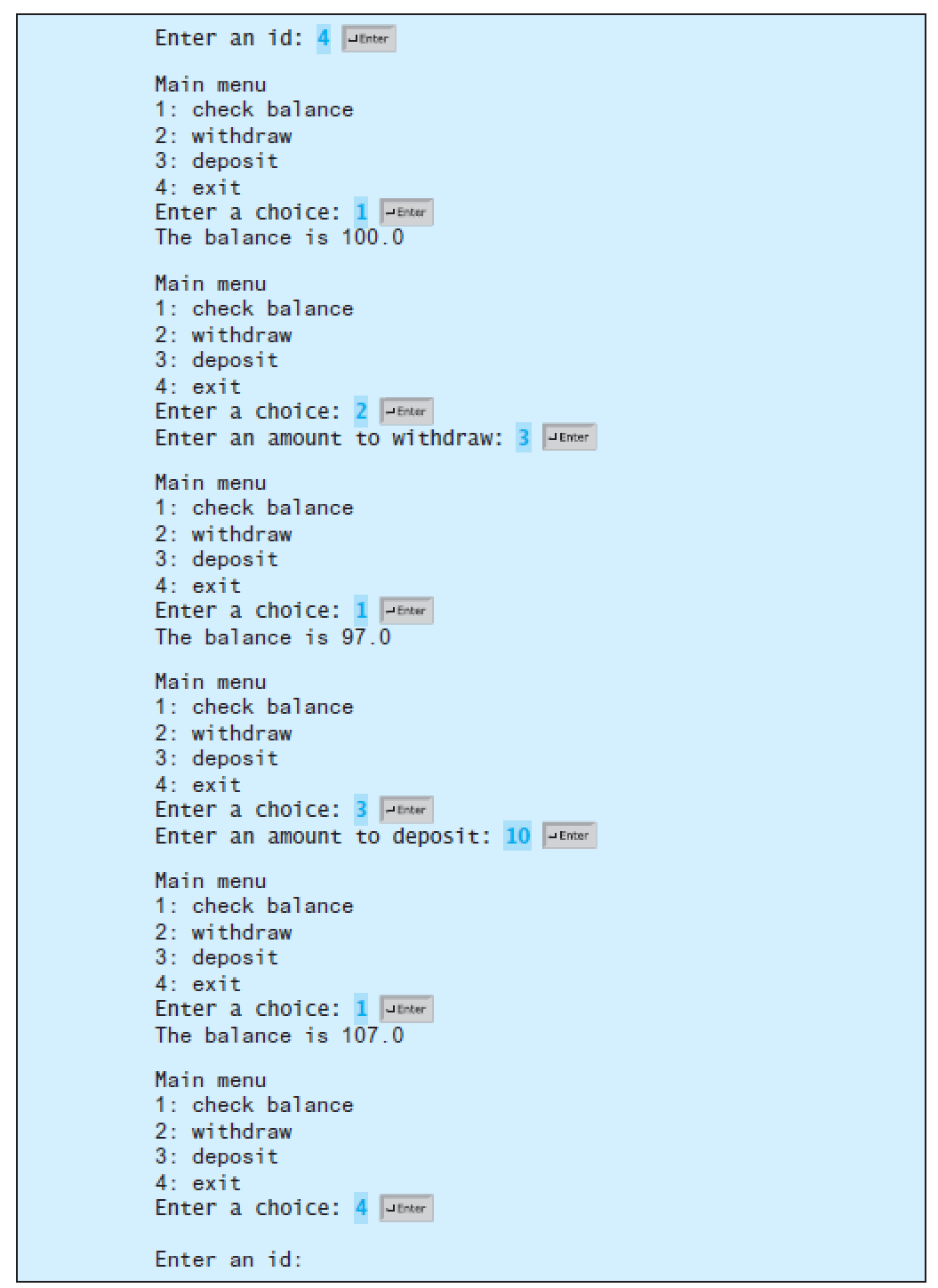
Monthly interest is **balance \* monthlyInterestRate**.

**monthlyInterestRate** is **annualInterestRate / 12**.

**annualInterestRate** is a percentage,for example 4.5%. You need to divide it by 100.

Draw the UML diagram for the class then implement the class.

Create another class (TestAccount) that usesthe **Account** class created above to simulate an ATM machine. Create 10 accounts in an array with id **0**, **1**, . . . , **9**, and an initial balance of $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter choice **1** for viewing the current balance, **2** for withdrawing money, **3** for depositing money, and **4** for exiting the main menu. Once you exit, the system will prompt for an id again. Thus, once the system starts, it will not stop.



|  |
| --- |
| **Account** |
| private id: int  private balance: double  private annualInterestRate: double; |
| Account()  Account(ID: int, bal: double)  setId(val: int) : void  getId(): int  setBalance(bal: double): void  getBalance(): double  setAnnualInterestRate(air: double): void  getAnnualInterestRate(): double  getMonthlyInterestRate(): double  getMonthlyInterest(): double  withdraw(): void  deposit(): void |

**Analysis:**

This question asks us to create a program which will prompt a user for an id. The id entered by the user allows you to access and modify a certain bank account. We can use a seperate method to create a sentinel loop which prompts the user with different menu options. The user must be able to check the balance in each account and each account must have a balance which is not connected to the other accounts. For this reason, it is best to use an array of objects to create all of the seperate accounts. The user must also be able to withdraw money from the account, therefore subtracting money from the balance. The user must be able to deposit money into the account, therefore adding money to the balance. Finally, the user must be able to exit from one account and enter another account. The call to the method which allows us to access the account must exist inside of an infinite loop. This allows the user to instantly be prompted to enter another id as soon as it exits from the previous account.

**Algorithm Design:**

1. Create an array of 10 Account objects.
2. Using a for loop to modify each object to be connected to an id and to have a starting balance of $100.
3. In a while loop prompt the user to enter an id.
4. If the id is not between 0 & 9 then it is invalid and not connected to an account
5. If it is between 0 & 9 then call the mainMenu(Account obj, int id) method
6. This will then enter the user into another while loop which will prompt them to select from the following menu options
7. 1: check balance will access the getBalance method from the account class which will return the balance.
8. 2: withdraw will access the withdraw method which will prompt the user to enter a value to subtracted from the balance.
9. 3: deposit will access the deposit method which will prompt the user to enter a value to be added to the balance.
10. 4: exit will change the while loops boolean condition exit to true which will then exit the while loop and return to the main method
11. The user will be prompted to enter another id

**Screenshots**

**Text

Description automatically generated**

**Text

Description automatically generated**