Course Project Specifications

Objectives:

- Create a video presentation discussing the NetBeans project that you created based on the requirements of the problem given.
- Apply the basic knowledge obtained from the modules of Object-Oriented Programming.
- Demonstrate the knowledge attained throughout the course by demonstrating the theories, problem solving, and applications as required by the activity.

Description Specifications:

- Create a video presentation explaining your solution to the problem given in this activity.
- Provide a detailed explanation of the classes and methods defined in the created Java program. Run your program in the NetBeans IDE and provide user inputs that will validate its compliance with the program specifications.
- The video should include yourself while presenting your solution. You need to be clearly visible in the video recording.
- Upload your video to Youtube or Vimeo.
- Upload the course project (**.rar or .zip** format) to the link provided in the LMS.
- The course project submission should be in a folder of compressed format (.rar or .zip) which includes the following:
  1. This MS Word Course Project template with the link of Youtube or Vimeo
  2. The NetBeans project folder
- The course project filename should be as follows:
  Course Project Filename: CS6203_CourseProject_<lastname_firstname>
  Example: **CS6203_CourseProject_Blanco_Maria**
- Be sure to create your own Java program and comply with the activity requirements. Non-compliance will require resubmission.
- The highest score for resubmission is 50%.

Paste the link of the video recording here:

Activity Objectives:

At the end of this activity, you should be able to:

- define classes and methods
- implement polymorphism and inheritance
- instantiate an object array
- apply Exception to all input validations
- create a Java application for multiple users

What to Prepare for the Activity:

- NetBeans IDE 12.0 or higher version
- JDK10 (Java Development Kit 10) or higher version

Procedure:

- Create a NetBeans project for this activity. The project name should be as follows:
  Project Name: CS6203_JavaCredit_<lastname_firstname>
  Example: **CS6203_JavaCredit_Blanco_Maria**

- The class names to be created are the following:
  - Client (the main class that contains the main method and the implementation of the main menu)
  - Credit (the class where attributes and methods are defined)

- All class names must be suffixed with your last name.
  - Client<your_last_name> Ex. **ClientBlanco**
  - Credit<your_last_name> Ex. **CreditBlanco**

- Note that the object to be instantiated in the main method is an object array.
  For example: CreditBlanco [ ] cb = new CreditBlanco [100];

- Only NetBeans project will be accepted.

```
Java Credit Main Menu
[1] New Credit Account
```

[2] Credit Inquiry
[3] Purchases
[4] Payment
[5] Close Credit Account
[7] Exit

Program Specifications:

1. New Credit Account
    - Input the name of the client.
    - Input the annual income and assign the credit limit as follows:

Annual Income	**Credit Limit**
200,000 and above but not greater than 300,000	30,000
300,000 and above but not greater than 500,000	50,000
500,000 and above	100,000

    - Generate and display a random four-digit credit account number.
    - Display also the credit limit. The **credit limit** is the maximum amount of credit.
    - An annual income below 200,000 is not eligible for the credit.

2. Credit Inquiry
    - Input the credit account number and validate.
    - If the credit account number is valid, display the credit account number, credit account name, credit limit and credit balance.
    - The **credit balance** is the outstanding balance or the total amount due.

3. Purchases
    - Input the credit account number and validate.
    - If the credit account number is valid, input the amount of purchases of not less than Php 1 and not greater than the credit limit and the allowable purchase amount.
    - Compute and update the credit balance and allowable purchase amount as follows:

        Credit balance+=amount of purchases
        Interest=3% of credit balance
        Credit balance+=interest
        Allowable purchase amount = credit limit – credit balance

        - The **allowable purchase amount** is the difference between the credit limit and the credit balance. The credit balance should not exceed the credit limit.

4. Payment
    - Input the credit account number and validate.
    - If the credit account number is valid, input the amount of payment of not less than Php 1 and not greater than the credit balance.
    - Compute and update the credit balance as follows:

        Credit balance – = amount of payment

5. Close Credit Account

- Input the credit account number and validate.
- If the credit account number is valid, confirm if the user wants to close the account.
- If confirmed by the user, he should be required to pay all the credit balance before closing the account.
- Otherwise, go back to the main menu.
- If the account has been closed, the credit account should not exist anymore.

6. Exit
   - Terminates the program

7. All input values must be validated and must be required for re-entry of data when invalid.
8. The use of Exception for all input validations is required.

GRADING CRITERIA:

1. **Code Structure and Organization (10 points)**
   - (2) Code is well-organized with appropriate indentation and consistent formatting.
   - (2) Proper use of comments and documentation to explain code logic and functionality.
   - (2) Modular approach with functions or classes where appropriate.
   - (2) Code follows naming conventions for variables, functions, and classes.
   - (2) Code is free from unnecessary or redundant lines.
2. **Functionality and Requirements (20 points)**
   - (5) Code successfully compiles without errors.
   - (5) Code fulfils all specified requirements and functions as expected.
   - (5) Handles edge cases and input validation effectively.
   - (5) Demonstrates a clear understanding of the problem and its solution.
3. **Algorithm Design and Efficiency (10 points)**
   - (5) Demonstrates a thoughtful algorithm design that efficiently solves the problem.
   - (5) Proper use of data structures and algorithms.
4. **Testing and Debugging (15 points)**
   - (5) Code passes all test cases and produces expected outputs.
   - (5) Effective error handling and error messages for unexpected inputs.
   - (5) Evidence of debugging process and problem-solving skills.
5. **Creativity and Innovation (5 points)**
   - (5) Demonstrates creativity in problem-solving or adds innovative features.
6. **Video Presentation (40 points)**
   - The presenter is visible in the video.
   - Clear and audible voice with appropriate pacing.
   - Logical organization of ideas and information.
   - Communicates ideas clearly and concisely.
   - Video and audio quality are clear and free from distractions.

Total Points: 100

Comments and Feedback: