Programming Language Paradigms
CS 350 - 040
Department of Physics and Computer Science
Medgar Evers College
Exam 2

Instructions:

- **The exam requires completing tasks in two of the three programming languages covered [C++, Ruby, Python] within 120 minutes; however, one of the chosen languages must be Python.**
- **You need to modify two of the accompanying '`main`' files in the Exam03 directory of your GitHub repository. You can only add the required code specified by each problem; including additional libraries and modules is prohibited.**
- **If you choose C++, the class of the first problem must inherit the *Object* class from '`Utils.h`'.**
- **The `to_str()` overriding task is for the C++. For Python and Ruby, override `__str__()` and `to_s()`, respectively.**
- **Completing the tasks in the third language will be extra credit.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating or failing to follow any rules will result in an automatic zero (0) for the exam.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, PRINT YOUR NAME AND THE DATE ON BOTH THIS SHEET AND THE BLUE BOOK

Name: _____ **Date:** _____

Grading

Problem	Maximum Points	Points Earned
1	8	
2	12	
Total	20	

1. **Define the class *BalanceManager* that contains**

   - **a private floating-point field named `_amount`.**
   - **a private integer class field named `_count` initialized to 0.**
   - **a public default constructor that assigns 0 to `_amount` and increments `_count` by 1.**
   - **a public overloaded constructor that takes a floating-point parameter, assigns the parameter to `_amount`, and increments `_count` by 1. If `_amount` is negative, it assigns 0 to `_amount`.**
   - **a public virtual Boolean method named `possible_deposit()` that takes a floating-point parameter and returns true.**
   - **a public virtual Boolean method named `possible_withdraw()` that takes a floating-point parameter and returns true.**
   - **a public virtual Boolean method named `deposit()` that takes a floating-point parameter. It adds the parameter to `_amount` and returns true if the parameter is not negative and `possible_deposit()` returns true; otherwise, it returns false.**
   - **a public virtual Boolean method named `withdraw()` that takes a floating-point parameter. It subtracts the parameter from `_amount` and returns true if the parameter is not negative and `possible_withdraw()` returns true; otherwise, it returns false.**
   - **a public getter method for `_amount` named `balance()`.**
   - **a public integer class method named `total_accounts()` that takes no parameters and returns `_count`.**
   - **a public overridden `to_str()` method that returns a string in the format**

     `Balance:` x `USD`

     where x is the values of `_amount` with 2 decimal points.

2. **Define the class *BalanceManagerWithDailyTurnOver* that inherits *BalanceManager* and contains**

   - **a private floating-point field named `_current`.**
   - **a private floating-point field named `_maximum`.**
   - **a public default constructor that assigns 0 to both `_amount` and `current`, and assigns 5000 to `_maximum`.**
   - **a public overloaded constructor that takes a floating-point parameter, assigns the parameter to `_amount`, assigns 0 to `_current`, and assigns 5000 to `_maximum`.**
   - **a public overloaded constructor that takes two floating-point parameters, assigns the first parameter to `_amount`, assigns 0 to `_current`, and assigns the second parameter to `_maximum`. If `_maximum` is negative, it assigns 5000 to `_maximum`.**
   - **a public Boolean method named `possible_transaction()` that takes a floating-point parameter. It returns true if the sum of `_current` and the parameter is less than or equal to `_maximum`; otherwise, it returns false.**
   - **a public overridden `possible_deposit()` method that returns a `possible_transaction()` caller with its parameter as the argument of the caller.**
   - **a public overridden `possible_withdraw()` method that returns a `possible_transaction()` caller with its parameter as the argument of the caller.**
   - **a public overridden `deposit()` method that adds its parameter to `_current` and returns true if a `deposit()` superclass caller using the parameter as its argument returns true; otherwise, it returns false.**
   - **a public overridden `withdraw()` method that adds its parameter to `_current` and returns true if a `withdraw()` superclass caller using the parameter as its argument returns true; otherwise, it returns false.**
   - **a public getter method for `_current` named `turnover()`.**
   - **a public setter method for `_maximum` named `adjustment()` that performs the assignment only if the parameter is greater than or equal to `_current`.**
   - **a public void method named `reset()` that takes no parameters and assigns 0 to `_current`.**
   - **a public overridden `to_str()` method that returns a string in the format**

     `Balance:` x `USD`
     `Turnover:` y `USD`
     `Limit:` z `USD`

     where x, y, and z are the values of `_amount`, `_current`, and `_maximum`, respectively, such that each has 2 decimal points.