CS102A Assignment 5

In this assignment, you are going to build an intelligent service robot for a bank. The robot can provide the following services to a customer.

1. **Create a bank account:** the robot helps the customer create a bank account with an initial balance 0.0 RMB. A customer has a unique social security number. Each social security number can be used to create only one account (meaning that each customer can only have one account).
2. **Query account balance:** the robot helps the customer check the remaining balance in his/her account. If the customer does not have an account, this operation will fail with an error message.
3. **Make a deposit:** the robot helps the customer make a deposit into his/her account. If the customer does not have an account, this operation will fail with an error message.
4. **Withdraw cash:** the robot helps the customer withdraw an amount of money from his/her account. The operation will fail in any of the following conditions: (1) the customer does not have an account, (2) the customer's bank account does not have sufficient balance, (3) the bank does not have sufficient cash.
5. **Provide advices on house mortgage loan:** if the customer wants to apply for a house mortgage loan from the bank, the robot should be able to help calculate the monthly payment based on (1) the loan amount and (2) the loan term (the yearly rate varies according to the loan term) according to the formula below. For example, if customer would like to apply for a 30-year loan (360 months) with the amount 3 million RMB, then he/she needs to pay 15,921.80 RMB every month.

$$monthly\ \ payment = P \times \frac{R \times (1+R)^N}{(1+R)^N - 1}$$

$$P\ \ is\ \ the\ \ loan\ \ amount$$

$$R\ \ is\ \ the\ \ montly\ \ rate\ \ \left(= \frac{yearly\ \ rate}{12}\right)$$

$$N\ \ is\ \ the\ \ number\ \ of\ \ months\ \ for\ \ the\ \ loan$$

Loan term (years)	Yearly rate (%)
1 – 2	4.35
3 – 5	4.75
5+	4.90

Assignment requirements:

1. Please design and implement a `Customer` class. The class should have the following fields and methods:

   - `firstName` (`String`): a valid first name can only contain English letters (no other characters or spaces are allowed).
   - `lastName` (`String`): the requirement is the same as `firstName`.
   - `gender` (of enum type `Gender`): the enum type `Gender` has only two enum constants: `MALE`, `FEMALE`.
   - `socialSecurityNumber (String)`: a valid social security number of a customer contains only eight digits and the first digit cannot be zero.
   - Setter and getter methods for each of the above four fields
   - A constructor `Customer(String firstName, String lastName, Gender gender, String ssn)`: this constructor helps construct a new `Customer` object using the values of the passed arguments. The first name and the last name should be transformed to the following form if originally they are not in the form: the first letter should be upper-case and all remaining letters should be lower-case.
   - A static method `checkName(String name)` that can be called to check if a string is a valid customer name or not. The method returns `true` if the input string represents a valid name and `false` otherwise.
   - A static method `formatName(String name)` that can be called to transform a valid name string to make its first letter upper-case and the remaining letters lower-case. The method returns the formatted string.
   - A static method `checkSSN(String ssn)` that can be called to check if a string represents a valid social security number or not. The method returns `true` if the input string represents a valid social security number and `false` otherwise.

2. Please design and implement a `BankAccount` class. The class should have the following fields and methods:
   - `owner` (of `Customer` type): the owner of the bank account
   - `balance (double)`: the amount of money left in the account
   - Getter and setter methods for each of the above two fields
   - A constructor `BankAccount(Customer customer, double balance)`: this constructor helps construct a new `BankAccount` object using the values of passed arguments.

3. Please design and implement a `BankService` class. The class should have the following fields and methods:
   - `availableCash (double)`: this field represents the amount of cash that the bank has for customers to withdraw.
   - `accounts (ArrayList<BankAccount>` type`)`: this field records the list of existing accounts created at the bank
   - `getAccount(Customer customer)`: this method finds the account of a customer and returns the reference to the account object. It returns `null` if no account is found for the customer.
   - `getAccount(String ssn)`: this method finds the account of a customer according to a given social security number and returns the reference to the account object. It returns `null` if no account is found for the customer.
   - `checkAccountBalance(String ssn)`: this method finds the account of a customer according to a given social security number and prints the balance of the account if found. If the account is not found, the method prints an error message.
   - `createAccount(String firstName, String lastName, char gender, String ssn)`: this method helps create an account for a customer. It prints a congrats message if the account is successfully created. If the customer already has an account, the method prints the account balance.
   - `makeDeposit(String ssn, double amount)`: this method makes a deposit into a bank account according the social security number of the

customer. It prints the original balance and the balance after deposit if the deposit is made successful. If the amount is a negative value, the method prints an error message.

- `withdraw(String ssn, double amount)`: this method withdraws an amount of money from an account according to the social security number of the customer. It prints the original balance and the balance after withdraw if the withdraw operation is successful. If there is no enough balance in the account or the bank does not have sufficient cash, the method prints an error message. If the amount is a negative value, the method also prints an error message.
- `calculateMonthlyPayment(double loanAmount, int years)`: this method is a static method that calculates the monthly payment for a customer who wants to apply for a house mortgage loan. It prints detailed calculation results to on the screen.
- A no-argument constructor that initializes the `availableCash` to 100,000 and the `accounts` to an empty list.

We provide the robot's `Main` class. You are not allowed to modify the `Main` class. You can run it to test your code.

Submission requirements:

1. Please submit four source files (.java files): `Gender.java`, `Customer.java`, `BankAccount.java`, `BankService.java`.
2. No Chinese characters are allowed to appear in your code.
3. No package included.
4. You should strictly follow our descriptions to define the fields and methods.
5. In this assignment, you can only use Scanner to get input. The output of your program should strictly follow our test cases.
6. Please submit your assignment on the SAKAI site of your lab section. Marks will be deducted if you submit later than the deadline. If you submit your assignment within 24 hours after the deadline (grace period), your score will be half of the score you could get if the submission was made before the deadline. Assignments submitted after the grace period will not be graded (meaning you will get a zero for the assignment).