Question 1:
Having a BankAccount class (see the UML class diagram), we would like to manage the customer accounts.

<table>
<thead>
<tr>
<th>BankAccount</th>
</tr>
</thead>
<tbody>
<tr>
<td>- accId : int</td>
</tr>
<tr>
<td>- customerName : string</td>
</tr>
<tr>
<td>- balance : double</td>
</tr>
<tr>
<td>- type : string</td>
</tr>
<tr>
<td>+ BankAccount(in id : int, in name : string)</td>
</tr>
<tr>
<td>+ display() : void</td>
</tr>
<tr>
<td>+ deposit(in amount : double) : void</td>
</tr>
<tr>
<td>+ withdraw(in amount : double) : void</td>
</tr>
<tr>
<td>+ hasGreaterBalance(in account : BankAccount) : BankAccount</td>
</tr>
<tr>
<td>+ getType() : string</td>
</tr>
<tr>
<td>+ getBalance() : double</td>
</tr>
</tbody>
</table>

A BankAccount has:
- **accId** as an account number, the customer **name** and a **balance**. A **type** is a field that defines if the account is a golden or a silver account. An account is golden if its balance is greater or equal to 100,000 SAR, otherwise it is a silver one. The type of the account changes according to the value of the balance, when we perform a deposit or a withdraw operation.

The BankAccount provides several methods:
- A **constructor** that has as input parameters the account number and the customer name. Each new account starts with a silver account and a zero balance.
- **display** displays the account information.
- **deposit** allows for depositing an amount in the account.
- **withdraw** allows for getting an amount from the account.

The type of the account may change according to the balance. If the account is silver, the customer cannot withdraw more than 5000 SAR, whereas for the golden account the limit is 10000 SAR.

- **hasGreaterBalance** compares two accounts and returns the account that has the greater balance.
- **getType** and **getBalance** returns resp. the type and the balance of the account

**Question**: Implement the class BankAccount
**Question 2:**

Using the BankAccount class (see the UML class diagram), we want to write a test class (TestBankAccount) that does the following:

It manages n accounts (n is entered by the user). It also performs for each account a deposit and a withdraw operation (the amount should be read for each operation). The program should display:

- the silver account that has the maximum balance.
- the number of golden account and their average balance.

**Question:** Write the Java TestBankAccount that covers the above-defined requirements.