Having a Bank class (see the UML class diagram), we would like to manage the customer accounts.

```
<table>
<thead>
<tr>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>- name : String</td>
</tr>
<tr>
<td>- accNum [ ] : int</td>
</tr>
<tr>
<td>- balance [ ] : double</td>
</tr>
<tr>
<td>- type [ ] : String</td>
</tr>
<tr>
<td>- current : int</td>
</tr>
<tr>
<td>+ Bank (bankName : String, size: int)</td>
</tr>
<tr>
<td>+ searchIndexOfAccNumber(accNb : int ) : int</td>
</tr>
<tr>
<td>+ deposit(accNb : int, amount : double) : void</td>
</tr>
<tr>
<td>+ withdraw(accNb : int, amount: double ): void</td>
</tr>
<tr>
<td>+ addNewAccount(accNb: int, bal : double ) : boolean</td>
</tr>
<tr>
<td>+ deleteAccount(accNb: int ) : boolean</td>
</tr>
<tr>
<td>+ display(typ: String) : void</td>
</tr>
<tr>
<td>+ accNumThatHasMaxBalance( void) : int</td>
</tr>
<tr>
<td>+ averageInBalanceForAccount(typ: String) : double</td>
</tr>
<tr>
<td>+ whichTypeHasMaxAccNumber(void) : String</td>
</tr>
<tr>
<td>+ minNumberOfAccount(Bank :b ) : Bank</td>
</tr>
<tr>
<td>+ totalBalanceInTheBank() : double</td>
</tr>
<tr>
<td>+ deleteAllAccountOfType(typ: String): void</td>
</tr>
<tr>
<td>+constructBranchOfBankforAllAccountOfType(typ: String) : Bank</td>
</tr>
<tr>
<td>+constructTwoBranches(Bank b1, Bank b2) : void</td>
</tr>
</tbody>
</table>
```

The attributes:

- **name**: name of the Bank.
- **accNum**: array containing the account number of accounts.
- **balance**: array containing the balance of accounts.
- **type**: array containing the type of accounts. A type of account is “golden” if the balance is greater than or equal 100,000 SR, otherwise it is a “silver” type. When we perform a deposit or withdraw operation the type of the account may change according to the value of balance.
- **current**: current number of accounts in the bank.
Methods:

- **Bank(bankName : String, size : int)**: each new account starts with zero balance.
- **searchIndexOfAccNumber(accNb : int)**: returns the index in array of the account number accNb, otherwise returns -1.
- **deposit(accNb : int, amount : double)**: allows for depositing an amount amount for the account number accNb.
- **withdraw(accNb : int, amount : double)**: allows for debiting an amount amount from the account number accNb.
- **addNewAccount(accNb: int, bal : double)**: allows for adding a new account to the Bank with account number accNb and balance bal. It returns true if the operation is done, otherwise it returns false.
- **deleteAccount(accNb: int)**: allows for deleting an account from the Bank with account number accNb. It returns true if the operation is done, otherwise it returns false.
- **display(typ: String)**: allows for displaying the information of all accounts that have type typ.
- **accNumThatHasMaxBalance( void)**: It returns the account number which has max balance.
- **averageInBalanceForAccount(typ: String)**: average in balances for all accounts in type "typ".
- **whichTypeHasMaxAccNumbers( void)**: checks which type (golden of silver) has more number of accounts.
- **minNumberOfAccount(Bank : b)**: compare the current bank and b and return the bank that has minimum number of account.
- **totalBalanceInTheBank()**: returns the total balance of all accounts in the bank.
- **deleteAllAccountsOfType(typ: String)**: to delete all account of type typ
- **constructBranchOfBankForAllAccountOfType(typ: String)**: This method returns a Bank Branch from the current Bank and put on it all the accounts in type "typ".

Question:

1- Write in Java the class **Bank**
2- Write a main to test all the methods in the class **Bank**

**Solution:**

Class Bank

```java
public class Bank {
    private String name;
    private String type[];
    private int accnum[];
    private double balance[];
    private int current;

    public Bank(String Bname, int size){
        name = Bname;
        type = new String[size];
        accnum = new int[size];
        balance = new double[size];
        current = 0;
    }

    public int searchIndex(int accNum1){
        for(int i=0;i<current;i++){
            if(accnum[i]==accNum1)
                return i;
        }
        return -1;
    }

    // Other methods...
}
```
public void deposit(int accnum1, double amount) {
    int index = searchIndex(accnum1);
    balance[index] += amount;
    if (balance[index] >= 100000)
        type[index] = "golden";
    else type[index] = "silver";
}

public void withdraw(int accnum1, double amount) {
    int index = searchIndex(accnum1);
    if (balance[index] >= amount) { // withdraw only when amount available
        balance[index] -= amount;
        if (balance[index] >= 100000)
            type[index] = "golden";
        else type[index] = "silver";
    }
}

public boolean addNewAcc(int accnum1, double balance1) {
    int index = searchIndex(accnum1);
    if (accnum.length <= current)
        return false;
    else{
        if (index != -1)
            return false;
        else{
            accnum[current] = accnum1;
            balance[current] = balance1;
            if (balance[current] >= 100000)
                type[current] = "golden";
            else type[current] = "silver";
            current++;
            return true;
        }
    }
}

public boolean deleteAcc(int accnum1) {
    int index = searchIndex(accnum1);
    if (index == -1)
        return false;
    else{
        for (int i = index; i < current; i++) { // shifting to delete the account;
            accnum[i] = accnum[i + 1];
            balance[i] = balance[i + 1];
            type[i] = type[i + 1];
        }
        current--;
        return true;
    }
}

public void display(String typ) {
    for (int i = 0; i < current; i++) {
        if (type[i].equals(typ)) {
            System.out.println("Account number : " + accnum[i]);
            System.out.println("Account balance: " + balance[i]);
            System.out.println("Account Type : " + type[i]);
            System.out.println("==========================");
        }
    }
}
public int accNumThatHasMaxBalance(){
    int maxindex = 0;
    for(int i=0;i<current;i++){
        if(balance[i]>balance[maxindex]){
            maxindex = i;
        }
    }
    return accnum[maxindex];
}

public double averageInBalanceForAccount(String typ){
    int sum = 0;
    int numberOfacc=0;
    if(current == 0) // to avoid dividing by 0
        return 0;
    for(int i=0;i<current;i++){
        if(type[i].equals(typ)){
            sum += balance[i];
            numberOfacc++;
        }
    }
    return sum/numberOfacc;
}

public String whichTypeHasMaxAccNumbers(){
    int numberOfGolden=0;
    int numberOfSilver=0;
    for(int i=0;i<current;i++){
        if(type[i].equals("golden")){
            numberOfGolden++;
        }else{
            numberOfSilver++;
        }
    }
    if(numberOfGolden>numberOfSilver)
        return "golden";
    else
        return "silver";
}

public Bank minNumberOfAccount(Bank b){
    if(current<b.current)
        return this;
    else return b;
}

public double totalBalanceInTheBank(){
    int sum = 0;
    for(int i=0;i<current;i++){
        sum += balance[i];
    }
    return sum;
}

public String nameOfBankThathasGreatestTotalBalance(Bank b){
    if(totalBalanceInTheBank() > b.totalBalanceInTheBank() )
        return name;
    else return b.name;
}
public void deleteAllAccountsOfType(String typ) {
    for (int i = 0; i < current; i++) {
        if (type[i].equals(typ)) {
            deleteAcc(accnum[i]);
            i--; // after shifting we have to check index i again
        }
    }
}

public Bank constructBranchOfBankForAllAccountOfType(String typ) {
    Bank newBranch = new Bank(name + " : " + typ + " branch", accnum.length);
    // creates a branch with type name and same size of the current bank
    for (int i = 0; i < current; i++) {
        if (type[i].equals(typ)) {
            newBranch.addNewAcc(accnum[i], balance[i]);
        }
    }
    return newBranch;
}

public void constructTwoBranches(Bank b1, Bank b2) {
    for (int i = 0; i < current; i++) {
        if (type[i].equals("golden")) {
            b1.addNewAcc(accnum[i], balance[i]);
        } else {
            b2.addNewAcc(accnum[i], balance[i]);
        }
    }
}
public class BankTest {

    public static void main(String[] args) {

        Bank ksuBank = new Bank("KSU Bank", 100);
        ksuBank.addNewAcc(1111, 100000);
        ksuBank.addNewAcc(2222, 50000);
        ksuBank.addNewAcc(3333, 10000);
        ksuBank.addNewAcc(4444, 500000);
        ksuBank.addNewAcc(5555, 300000);
        ksuBank.display("golden");
        System.out.println("++++++++++++++++++++++++++++++++++++++++");
        ksuBank.deposit(2222, 70000);
        ksuBank.withdraw(1111, 100);
        ksuBank.deleteAllAccountsOfType("silver");
        System.out.println("All silver accounts deleted....");
        ksuBank.display("golden");
        ksuBank.display("silver");
        ksuBank.addNewAcc(6666, 10000);
        System.out.println("The account number which has max balance:" +
                ksuBank.accNumThatHasMaxBalance());
        System.out.println("The average of balance in golden accounts:" +
                ksuBank.averageInBalanceForAccount("golden"));
        System.out.println("The type which has max account number:" +
                ksuBank.whichTypeHasMaxAccNumbers());

        Bank otherBank = new Bank("Other Bank", 100);
        otherBank.addNewAcc(7777, 90000);
        System.out.println("Total balance of KSU Bank is :" +
                ksuBank.totalBalanceInTheBank());
        System.out.println("Total balance of the Other Bank is:" +
                otherBank.totalBalanceInTheBank());
        System.out.println("Name of the bank that has greatest total balance:" +
                ksuBank.nameOfBankThatHasGreatestTotalBalance(otherBank));

        Bank ksuGolden = new Bank("KSU golden branch", 100);
        Bank ksuSilver = new Bank("KSU Silver branch", 100);
        ksuBank.constructTwoBranches(ksuGolden, ksuSilver);
        System.out.println("The accounts in the golden branch of KSU Bank are:");
        ksuGolden.display("golden");
        ksuGolden.display("silver"); // shouldn't display anything because all 
                //accounts in the branch are golden !!!!
        System.out.println("The accounts in the silver branch of KSU Bank are:");
        ksuSilver.display("golden"); // shouldn't display anything because all 
                //accounts in the branch are silver !!!!
        ksuSilver.display("silver");
    }
}