**Department of computer science and engineering**

**HW #3**

**Question** 1 : Extend Methode

**Usage** : in hash Table With Linear Probing

**When to be called** : When table is filled up with elements **Code :**

public void expandTheTableWhenFilled() { int newSize = ArrayOfAccount.length \* 2; //getting the new size

Account refrenceToOldArray[] = ArrayOfAccount; //haviing a refrence to not lose items or we could have used arrayOfAccount.clone()

ArrayOfAccount = new Account[newSize]; //creating the new array counter = 0; //so that we have new array so counter start from the new array

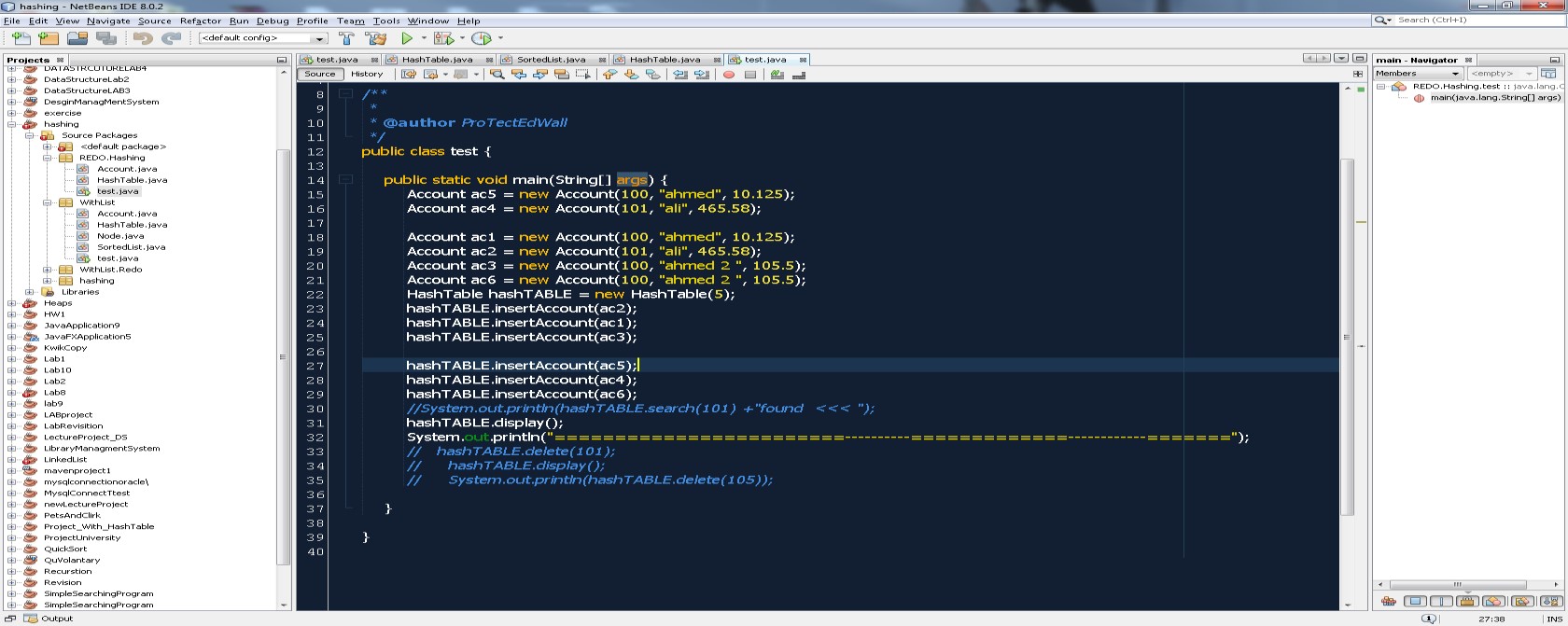
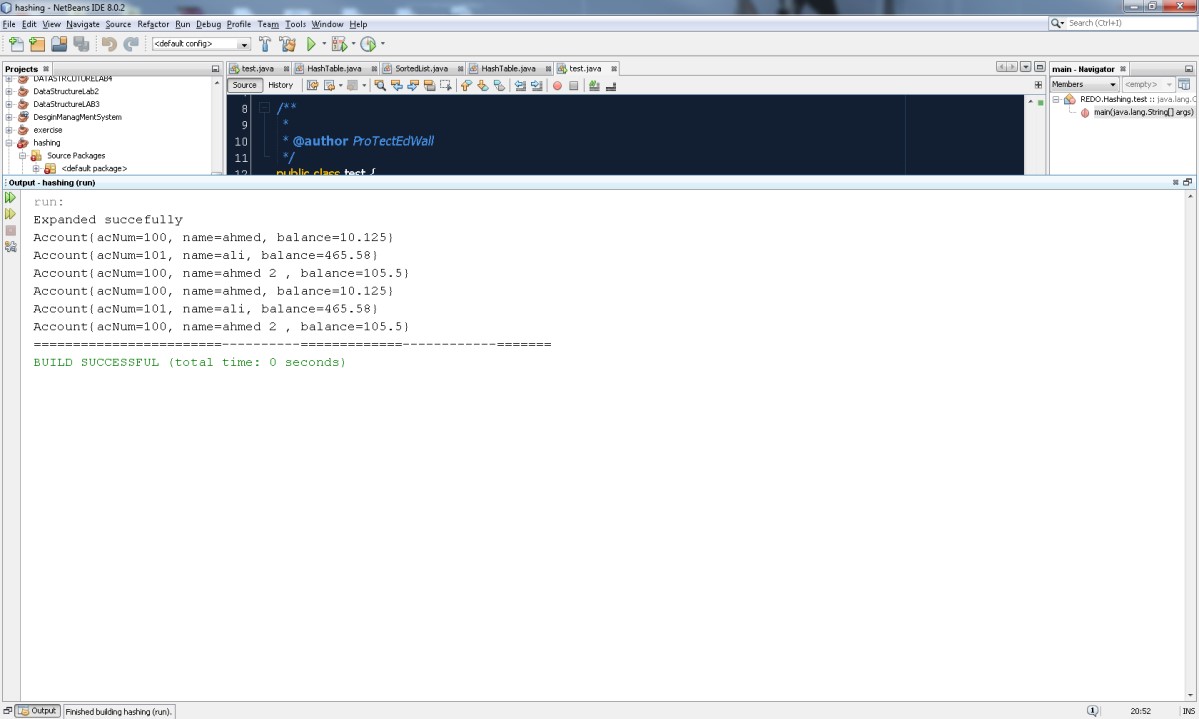
//now i will loop to rehash and insert items to the bigger array for (int i = 0; i < refrenceToOldArray.length; i++) { if (refrenceToOldArray[i] != null) { if (refrenceToOldArray[i].getAcNum() != -1) { insertAccount(refrenceToOldArray[i]); counter++;

}

}

}}

 **Screenshot of data to be inserted**



**Question** 2: Duplicate Methode

**Usage** : in hash Table With separate chaining

**When to be called** : When average number of nodes bigger than user input parameter **Code :**

public void duplicate(int max) {

if (getAllAverage() > max) {

//linked list exist here

int newSize = hashArray.length \* 2; //setting new size for the array

SortedList arr[] = hashArray;//gtting refrence from the old array

hashArray = new SortedList[newSize];

for (SortedList list2 : arr) {//copy the elements and rehash to the new array if (list2 != null) {

for (Node mynode : list2.getArrayListFromAllItems()) {//getting list from each sortedList

if (mynode != null) {

insert(mynode.data);//inserting the data into the new array and it will be automatically hashed // System.out.println("how many items inserted");

}

}

}

//now we have to get all nodes in that list and multiplay the sie of the arry by two

}

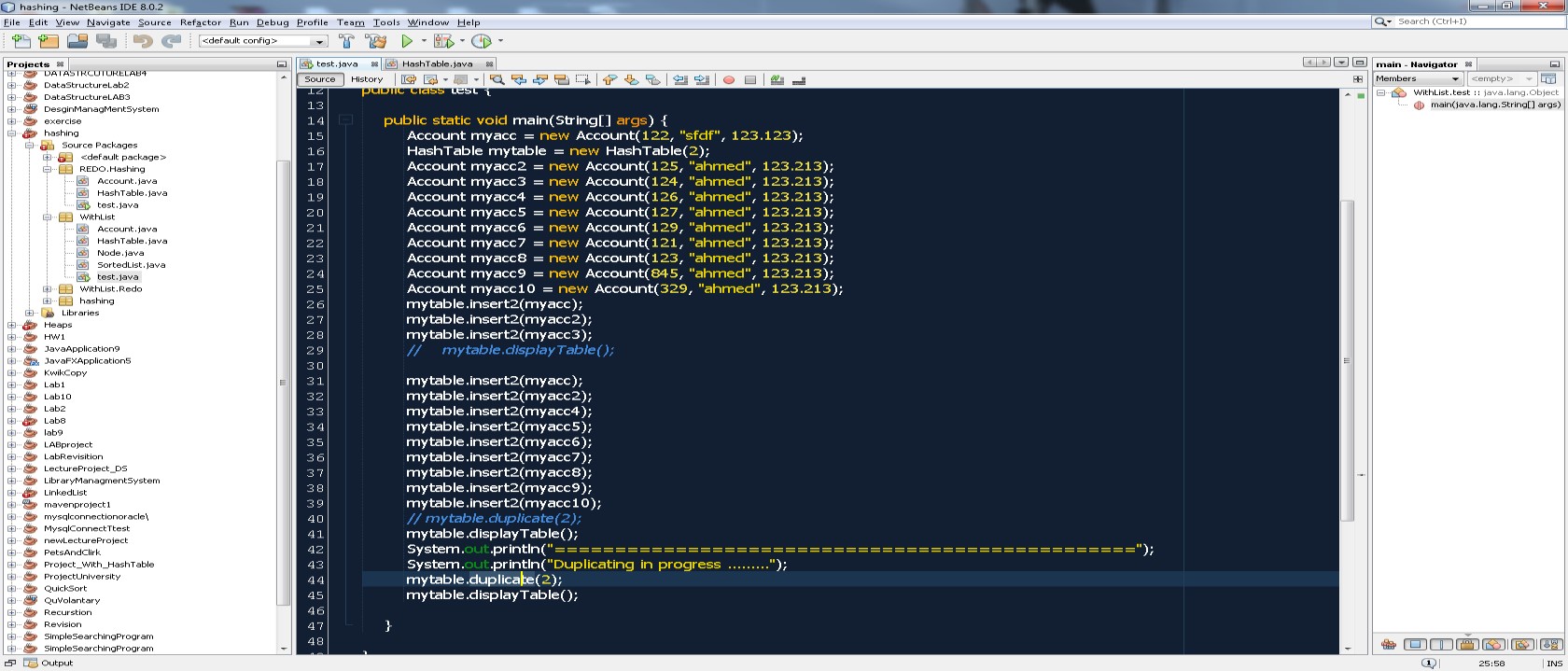
System.out.println("The system has been duplicated succefully ");

System.out.println("=================================");

}

}





Supporting methods :

1. public double getAllAverage() {

// System.out.println("in getallevergae "); return getHowmanyElementsInallTable() / hashArray.length;

}

1. public int getHowmanyElementsInallTable() {

int tempCounter = 0;

for (int i = 0; i < hashArray.length; i++) {

if (hashArray[i] != null) {

tempCounter += hashArray[i].sizeOfList();

}

}

System.out.println("returning gethowmanyelements" + tempCounter); return tempCounter;

}

1. public int sizeOfList() {

Node current = first; int temoCounter = 0; while (current != null) { temoCounter++;

current = current.next;

}

System.out.println("size of list is " + temoCounter); return temoCounter;

}

1. public void insertAccount(Account account) {

int hashValue = hashFunction(account.getAcNum());

if (isFull()) {

expandTheTableWhenFilled();

System.out.println("Expanded succefully"); insertAccount(account);

} else {

if (ArrayOfAccount[hashValue] == null) // OK , now we can insert

{

ArrayOfAccount[hashValue] = account;

} else { //if not then continue looping until you find a place , we will use the linear propping here ya3ni One by One.... int flag\_of\_the\_place = 0;

while (true) {

hashValue = hashFunction(hashValue + 1) % ArrayOfAccount.length; if (ArrayOfAccount[hashValue] == null && flag\_of\_the\_place <

ArrayOfAccount.length - 1) {

ArrayOfAccount[hashValue] = account;

counter++; //increase numbver of element in the array return;

}

flag\_of\_the\_place++;

}

}

}

}

5- public boolean isFull() { //return true if the array is full hashtable array if (counter == ArrayOfAccount.length - 1) { return true; } else { return false;

}

}