```java
//CHRIS TUFENKJIAN
//COMP282 ASSIGNMENT 1
import java.util.TreeMap;

public  class Table <K extends Comparable<K>, T>  //K = key,  T = item
{                                                                    //Extends
comparable is there if you are planning to compare that item
      TreeMap<K,T> tree    = new TreeMap<K,T>();  //Instantiate trees
      TreeMap<K,T> treeclone = new TreeMap<K,T>();

  public boolean isEmpty()
  {
      if (tree.size()>0)                                    //if size is greater than 0
then table is not empty, returns false
            return false;
      return true;
  }

   public int size()                                    //simple method to check tree size
using the size() function
   {
         return tree.size();
   }

   public void tableInsert(K key, T item)       throws TableException //insert method
with exception
   {
         if(tree.get(key) != null)
               throw new TableException
                              ("ERROR, THIS KEY ALREADY EXISTS.");
         else
               tree.put(key, item);
   }

   public boolean tableDelete(K key)            //simple delete table method using the
remove() function
   {
         if (tree.size() > 0)
         {
               tree.remove(key);
               return true;
         }
         else return false;
   };

   public T tableRetrieve(K key)
   {
         return tree.get(key);
   }

@SuppressWarnings("unchecked")
public void printTable()                                //method to print toString in
student object,pollfirst entry
   {                                                            //empties the tree while
retrieving, hence a copy of the tree must be made
         treeclone = (TreeMap<K, T>)tree.clone();//so no data is lost
         int n = treeclone.size();
         for (int i = 0; i<n; i++)
         {
              System.out.println(treeclone.pollFirstEntry());
         }
   }
}
```

```java
public class Student
{
	public	int	unitsearned;
	public	int	studentID; //student key
	public	String	name;
	public	String	major;

	public Student(int id, String Name, int UnitsEarned, String Major)
	{
		this.studentID = id;
		this.name = Name;
		this.major = Major;
		this.unitsearned = UnitsEarned;
	}

  @Override
  public String toString() //OVERRIDES DEFAULT TOSTRING METHOD AND RETURNS CUSTOM
  {
      String s = "Name: " + name
                  + " *** Student ID: " + studentID
                  + " *** Major: " + major
                  + " *** UnitsEarned" + unitsearned;
    return s;
  }

}
```

```java
@SuppressWarnings("serial")
public class TableException extends Exception
{
	public TableException (String message)
	{
		super(message);
	}

}
```

```
public class TableTest
{
	public static void main(String[] args) throws TableException
	{
		Table<Integer,Student> table = new Table<Integer, Student>(); //When creating
table, you must specify the types it will contain.

		Student Bob   = new Student(1001,"Bob",  110,"Cheese Appreciation Studies");
		Student Joe   = new Student(1002,"Joe",  134,"Economics");
		Student Jaime = new Student(1003,"Jaime",150,"Anthrobiology");
		Student Robb  = new Student(1004,"Robb", 99, "Emotional Character studies");
		Student Cathy = new Student(1005,"Cathy",85, "Women Studies");
		Student Stan  = new Student(1006,"Stan", 200,"Masters in City Urban
Planning");
		Student Bran  = new Student(1007,"Bran", 80, "PHD in Library Studies");
		Student Davos = new Student(1008,"Davos",90, "PHD In Car Washing");
		Student Arya  = new Student(1009,"Arya", 95, "Sword Fighting");
		Student John  = new Student(1010,"Jon",  110,"Snow Studies");

		System.out.println("TESTING DELETE STUDENT 1001: " +
table.tableDelete(1001));
		System.out.println("ORIGINAL TREE SIZE: " + table.size());
		table.tableInsert(Bob.studentID,Bob);
		table.tableInsert(Joe.studentID,Joe);
		table.tableInsert(Jaime.studentID,Jaime);
		System.out.println("AFTER ADDING STUDENT ONE, TWO, THREE, TREE SIZE: " +
table.size());
		System.out.println("TESTING DELETE STUDENT 1001: " +
table.tableDelete(1001));
		System.out.println("AFTER REMOVING STUDENT ONE, TREE SIZE: " + table.size());
		System.out.println("RETRIVE VALUE FROM KEY 1002: " +
table.tableRetrieve(1002));
		System.out.println("RETRIVE VALUE FROM KEY 1012 (DOES NOT EXIST): " +
table.tableRetrieve(1012));
		table.tableInsert(Robb.studentID,Robb);
		table.tableInsert(Cathy.studentID,Cathy);
		table.tableInsert(Stan.studentID,Stan);
		table.tableInsert(Bran.studentID,Bran);
		System.out.println("ADDING four,five, six, seven:");
		try
		{
			table.tableInsert(Bran.studentID,Bran);
		}
		catch(TableException e)
		{
			System.out.println(e);
		}
		System.out.println("WHAT IS THE TREE SIZE: " + table.size());
		System.out.println("RETRIVE VALUE FROM KEY 1003: " +
table.tableRetrieve(1003));
		System.out.println("ADDING eight,nine, ten:");
		table.tableInsert(Davos.studentID,Davos);
		table.tableInsert(Arya.studentID,Arya);
		table.tableInsert(John.studentID,John);
		System.out.println("RETRIEVE TABLE TWO, NOT THE ADDRESS:" +
table.tableRetrieve(1002).name);
		table.printTable();

		System.out.print("CHECK THAT SIZE IS SAME AFTER POLLFIRSTENTRY: " +
table.size());

	}
}
```

## OUTPUT:

```
TESTING DELETE STUDENT 1001: false
ORIGINAL TREE SIZE: 0
AFTER ADDING STUDENT ONE, TWO, THREE, TREE SIZE: 3
TESTING DELETE STUDENT 1001: true
AFTER REMOVING STUDENT ONE, TREE SIZE: 2
RETRIVE VALUE FROM KEY 1002: Name: Joe *** Student ID: 1002 *** Major: Economics ***
UnitsEarned134
RETRIVE VALUE FROM KEY 1012 (DOES NOT EXIST): null
ADDING four,five, six, seven:
TableException: ERROR, THIS KEY ALREADY EXISTS.
WHAT IS THE TREE SIZE: 6
RETRIVE VALUE FROM KEY 1003: Name: Jaime *** Student ID: 1003 *** Major: Anthrobiology
*** UnitsEarned150
ADDING eight,nine, ten:
RETRIEVE TABLE TWO, NOT THE ADDRESS:Joe
1002=Name: Joe *** Student ID: 1002 *** Major: Economics *** UnitsEarned134
1003=Name: Jaime *** Student ID: 1003 *** Major: Anthrobiology *** UnitsEarned150
1004=Name: Robb *** Student ID: 1004 *** Major: Emotional Character studies ***
UnitsEarned99
1005=Name: Cathy *** Student ID: 1005 *** Major: Women Studies *** UnitsEarned85
1006=Name: Stan *** Student ID: 1006 *** Major: Masters in City Urban Planning ***
UnitsEarned200
1007=Name: Bran *** Student ID: 1007 *** Major: PHD in Library Studies *** UnitsEarned80
1008=Name: Davos *** Student ID: 1008 *** Major: PHD In Car Washing *** UnitsEarned90
1009=Name: Arya *** Student ID: 1009 *** Major: Sword Fighting *** UnitsEarned95
1010=Name: Jon *** Student ID: 1010 *** Major: Snow Studies *** UnitsEarned110
CHECK THAT SIZE IS SAME AFTER POLLFIRSTENTRY: 9
```