**자료구조(박요한 교수님)**

**TreeMap를 사용한 Member 삽입 삭제 출력**

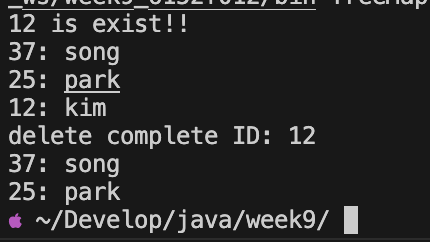
컴퓨터공학과 유상현

**1.** **프로그램 소스코드**

App.java

|  |
| --- |
| package TreeMap;  import java.util.\*;  class MemberIDCompare implements Comparator<Integer> {  @Override  public int compare(Integer o1, Integer o2){  return o2.intValue() - o1.intValue(); // ID 오름차순  }  }  class Member {  private int ID;  private String name;  Member (int ID, String name){  this.ID = ID;  this.name = name;  }  public int getID(){  return this.ID;  }  public String getName(){  return this.name;  }  }  class Program{  private TreeMap<Integer, Member> treeMap;  Program(){  treeMap = new TreeMap<Integer, Member>(new MemberIDCompare());  }  public void addMember(Integer id, Member input){  Set<Integer> ks = treeMap.keySet();  for(Iterator<Integer> itr = ks.iterator(); itr.hasNext(); ){  Member tmp = treeMap.get(itr.next());  if(tmp.getID() == id){  System.out.println(id + " is exist!!");  return ;  }  }  treeMap.put(id, input);  }  public void removeMember(int id){  Set<Integer> ks = treeMap.keySet();  for(Iterator<Integer> itr = ks.iterator(); itr.hasNext(); ){  Member tmp = treeMap.get(itr.next());  if(tmp.getID() == id){  treeMap.remove(id);  System.out.println("delete complete ID: " + id);  return ;  }  }  System.out.println("ID " + id+ " is not exist!!");  }  public void printAll(){  Set<Integer> ks = treeMap.keySet();  for(Iterator<Integer> itr = ks.iterator(); itr.hasNext(); ){  Member tmp = treeMap.get(itr.next());  System.out.println(tmp.getID() + ": "+ tmp.getName());  }  }  }  public class App {  public static void main(String[] args) {  Program program = new Program();  program.addMember(12, new Member(12, "kim"));  program.addMember(12, new Member(12, "you"));  program.addMember(25, new Member(25, "park"));  program.addMember(37, new Member(37, "song"));  program.printAll();  program.removeMember(12);  program.printAll();  }  } |

**2. 실행 화면**



**3. 느낀 점(고찰)**

treeMap 컬랙션을 이용하여 멤버 관리 프로그램을 만들어 보았는데 키와 값을 이용하여 값을 관리할 수 있다는 것이 시간 복잡도 측면에서 굉장히 편리하게 다가왔습니다. 다만 중복된 값을 체크할 때 해쉬를 사용하지 못해 모든 키값을 확인해야하는 것이 시간복잡도 측면에서 아쉬웠습니다.