```java
package uebung061;

import java.util.ArrayList;
import java.util.Iterator;

public interface MultiSet<T> extends Iterable<T> {
    //aufgabe a)
    //Regenwetter als Menge: { r,e,g,w,t,r}
    //als Multimenge: {(r,2), (e,4), (g,1), (w,1), (n,1), (t,2)}

    //aufgabe b)
    void add(T element);
    int count(T element);


}

```

1/1

2/2

```java
package uebung061;

import java.util.HashMap;
import java.util.Iterator;

public class HashMapMultiSet<T> implements MultiSet<T> {

    private HashMap<T, Integer> map;
    public HashMapMultiSet(){
        map = new HashMap();
    }

    public void add(T element){
        Integer count = map.get(element);
        if (count == null) {
            map.put(element, 1);
        } else {
            map.put(element, count + 1);
        }
    }

    public int count(T element){
        Integer count = map.get(element);
        if(count == null){
            return 0;
        }
        else{
            return count;
        }
    }

    public Iterator<T> iterator(){
        return map.keySet().iterator();
    }
}

```

A1 8/8

```java
package uebung062;

public class Aufgabe_2 {
    /*
    Zur Laufzeit ersetzt der Compiler T
    durch seine Beschränkung (oder durch Obejct)
    und parametrisiert Typen durch Basisstyspen.
    Zur Lafzeit sind die Exceptions dan genau gleich,
    da zurLaufzeit nicht zwischen INteger oder Double unterscheidet,
     weshalb die Fehlermeldung kommt.
     */
}

```

Genau, Type Erasure

2/2

```java
package uebung063;

import java.util.Collection;

public interface Company {

    /**
     * Adds a new employee to the company.
     * @param id The id that uniquely identifies the employee
     * @param name The name of the employee
     * @throws DuplicateIdException If the given id is already in use
     */
    void addEmployee(int id, String name) throws DuplicateIdException;

    /**
     * Returns the name of an employee identified by the given id,
     * or null if the id is not known
     * @param id The id of the employee
     * @return The name of the employee or null if the id is unknown
     */
    String getEmployeeName(int id);

    /**
     * Adds a new project to the company.
     * @param id The id that uniquely identifies the project
     * @param name The name of the project
     * @throws DuplicateIdException If the given id is already in use
     */
    void addProject(int id, String name) throws DuplicateIdException;

    /**
     * Returns the name of a project identified by the given id,
```

```
     * or null if the id is not known
     * @param id The id of the project
     * @return The name of the project or null if the id is unknown
     */
    String getProjectName(int id);

    /**
     * Assigns an employee to a project. An employee can be assigned to multiple projects,
     * but not to a single project twice (ignore duplicate assignments).
     * @param employeeId The id of the employee
     * @param projectId The id of the project
     * @throws UnknownIdException If either the employee or the project id is unknown
     */
    void assignEmployeeToProject(int employeeId, int projectId) throws UnknownIdException;

    /**
     * Removes an assigned employee from a project. Does nothing if the employee was
     * not assigned to the given project in the first place.
     * @param employeeId The id of the employee
     * @param projectId The id of the project
     * @throws UnknownIdException If either the employee or the project id is unknown
     */
    void removeEmployeeFromProject(int employeeId, int projectId) throws UnknownIdException;

    /**
     * Returns an ordered collection of employee ids, sorted by the name of the employees
     * @return A list of employee ids, sorted by name
     */
    Collection<Integer> getEmployees();

```

```
    /**
     * Returns an ordered collection of project ids that a given employee is assigned to,
     * sorted by the name of the projects
     * @param employeeId The id of the employee
     * @return A list of project ids for the given employee, sorted by name
     * @throws UnknownIdException If the employee id is unknown
     */
    Collection<Integer> getProjectsForEmployee(int employeeId) throws UnknownIdException;

}

```

```java
package uebung063;

public class StarkTest {

    public static void main(String[] args) {
        Company stark = new StarkEnterprises();
        try {
            stark.addEmployee(0, "Tony");
            stark.addEmployee(1, "Pepper");
            stark.addEmployee(2, "Jarvis");
            stark.addProject(0, "Suit");
            stark.addProject(1, "Jarvis");
            stark.addProject(2, "Jarvis");
            stark.addProject(3, "Finances");
            stark.assignEmployeeToProject(0, 0);
            stark.assignEmployeeToProject(0, 1);
            stark.assignEmployeeToProject(1, 3);
            stark.assignEmployeeToProject(2, 0);
            stark.assignEmployeeToProject(2, 2);
            System.out.println(stark);
        } catch (InvalidIdException e) {
            //Warum funktioniert e.getID nicht???
            System.out.println("Invalid ID: " + e.getId());
        }
        catch(DuplicateIdException e){
            System.out.println("Duplicate ID: " + e.getId());
        }
        catch(UnknownIdException e){
            System.out.println("Unknown ID: " + e.getId());
        }
    }

}
```

weil getID in InvalidIdException implementiert sein muss

```java
package uebung063;

import java.util.*;

public class StarkEnterprises implements Company{

    private Map<Integer, String> projects;
    private Map<Integer, String> employees;
    private Map<Integer, Integer> employeeproject;

    @Override
    public void addEmployee(int id, String name) throws DuplicateIdException {
        if(employees.containsKey(id)){
            throw new DuplicateIdException();
        }
        employees.put(id, name);

    }

    @Override
    public String getEmployeeName(int id) {
        if(employees.get(id) != null){
            return employees.get(id);
        }
        return null;

    }

    @Override
    public void addProject(int id, String name) throws DuplicateIdException {
        if(projects.containsKey(id)){
            throw new DuplicateIdException();
        }
        projects.put(id, name);

    }

    @Override
    public String getProjectName(int id) {
        StringBuilder sb = new StringBuilder();
        List<Integer> sortedEmployees = new ArrayList<>(employees.keySet());
```

ein employee kann in mehreren Projekten sein (aber nicht in einem mehrmals) -0,5P

ist das hier nicht die toString Methode? ProjectName muss doch nur projects.get(id) zurückgeben -0,5

```
        sortedEmployees.sort(Comparator.comparing(employees::get));
        for (int employeeId : sortedEmployees) {
            String employeeName = employees.get(employeeId);
            List<Integer> sortedProjects = new ArrayList<>(employeeproject.getOrDefault(employeeId, Collections.emptySet()));
            sortedProjects.sort(Comparator.comparing(projects::get));
            sb.append(employeeName).append("[").append(employeeId).append("]: ");
            for (int projectId : sortedProjects) {
                String projectName = projects.get(projectId);
                sb.append(projectName).append("[").append(projectId).append("] ");
            }
            sb.append("\n");
        }
        return sb.toString();
    }

    @Override
    public void assignEmployeeToProject(int employeeId, int projectId) throws UnknownIdException {
        if (!employees.containsKey(employeeId) || !projects.containsKey(projectId)) {
            throw new UnknownIdException();
        }
        employeeproject.put(employeeId, projectId);
    }

    @Override
    public void removeEmployeeFromProject(int employeeId, int projectId) throws UnknownIdException {
        if (!employees.containsKey(employeeId) || !projects.containsKey(projectId)) {
            throw new UnknownIdException();
```

```java
        }
        employeeproject.remove(employeeId, projectId);
    }

    @Override
    public Collection<Integer> getEmployees() {
        List<Integer> sortedEmployees = new ArrayList<>(employees.keySet());
        sortedEmployees.sort(new EmployeeComparator(employees));
        return sortedEmployees;

    }

    @Override
    public Collection<Integer> getProjectsForEmployee(int employeeId) throws UnknownIdException {
        if (!employees.containsKey(employeeId)) {
            throw new UnknownIdException();
        }

        //WARUM GEHT DAS HIER NICHT??
        List<Integer> projectsList = employeeproject.get(employeeId);
        if (projectsList == null) {
            projectsList = new ArrayList<Integer>();
        }


        projectsList.sort(new ProjectComparator(projects));


        List<Integer> resultList = new ArrayList<>(projectsList);

        return resultList;
    }
```

weil employeeproject keine Listen sondern nur Integer enthält (siehe Kommentar am Anfang)

```
}
```

```java
package uebung063;

import java.util.Comparator;
import java.util.Map;

public class ProjectComparator implements Comparator<Integer> {
    private Map<Integer, String> employees;

    public ProjectComparator(Map<Integer, String> employees) {
        this.employees = employees;
    }

    @Override
    public int compare(Integer id1, Integer id2) {
        String string1 = employees.get(id1);
        String string2 = employees.get(id2);
        int result = string1.compareTo(string2);
        if (result == 0) {
            result = id1.compareTo(id2);
        }
        return result;
    }
}
```

achtung: theoretisch null für string1 und/oder string2 möglich, wenn beide null, sind sie auch gleich aber string1.compareTo geht womöglich nicht

```java
package uebung063;

import java.util.Comparator;
import java.util.Map;

public class EmployeeComparator implements Comparator<Integer> {
    private Map<Integer, String> employees;

    public EmployeeComparator(Map<Integer, String> employees) {
        this.employees = employees;
    }

    @Override
    public int compare(Integer id1, Integer id2) {
        String string1 = employees.get(id1);
        String string2 = employees.get(id2);
        int result = string1.compareTo(string2);
        if (result == 0) {
            result = id1.compareTo(id2);
        }
        return result;
    }
}

```

```java
package uebung063;

public class InvalidIdException extends Exception{
}

```

```java
package uebung063;

public class UnknownIdException extends Exception {

}

```

```java
package uebung063;

public class DuplicateIdException extends Exception
{
}

```

9/10