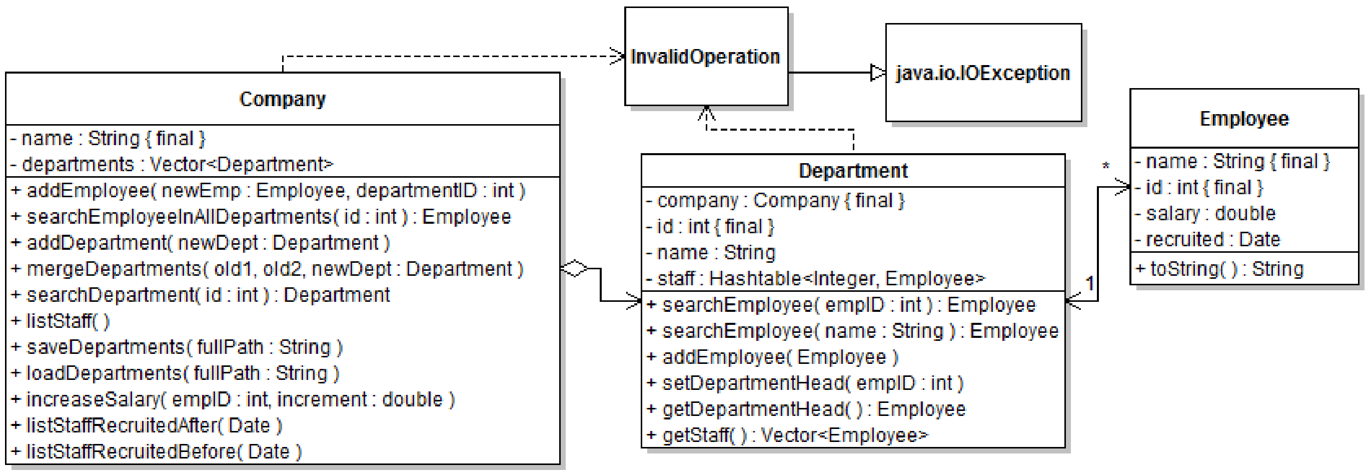
BLM2562 OBJECT ORIENTED PROG., Section 2, 2nd MIDTERM **(11/05/2015)**

Duration:	90mins.		**Score:**				**Student Nr:**	**Signature:**
Grading:	**1**	**2**	**3**	**4**	**5**	**6**	**Name, Surname:**	
	10	35	35	20				

QUESTIONS



InvalidOperation
java.io.IOException
Company
- name : String { final }
- departments : Vector<Department>
+ addEmployee(newEmp : Employee, departmentID : int)
+ searchEmployeeInAllDepartments(id : int) : Employee
+ addDepartment(newDept : Department)
+ mergeDepartments(old1, old2, newDept : Department)
+ searchDepartment(id : int) : Department
+ listStaff( )
+ saveDepartments(fullPath : String)
+ loadDepartments(fullPath : String)
+ increaseSalary(empID : int, increment : double)
+ listStaffRecruitedAfter(Date)
+ listStaffRecruitedBefore(Date)
Department
- company : Company { final }
- id : int { final }
- name : String
- staff : Hashtable<Integer, Employee>
+ searchEmployee(empID : int) : Employee
+ searchEmployee(name : String) : Employee
+ addEmployee(Employee)
+ setDepartmentHead(empID : int)
+ getDepartmentHead( ) : Employee
+ getStaff( ) : Vector<Employee>
*
1
Employee
- name : String { final }
- id : int { final }
- salary : double
- recruited : Date
+ toString( ) : String

Answer these questions according to the UML class schema given above. You may need to extract hidden information from the schema and add necessary code while answering the questions.

Question 1: Write the source code of the toString method of class Employee. This method returns a String that contains all information about the employee excluding salary, including the department. All information will be separated by a tab character. If this employee is the head of that department, it must be also included in the String.

Question 2: Write the source code of class Department. Details of some of its methods are as follows:

- addEmployee: Before adding the employee to this department, this method must search the employee in the entire company and generate an InvalidOperation exception if that employee has already been employed.
- setDepartmentHead: The head of a department must be a member of that department. An attempt to break this rule must be prevented by generating an InvalidOperation exception.

Question 3: Write the source code of only the following methods of class Company. The details are as follows:

- mergeDepartments: This method merges the departments old1 and old2 under the department newDept. Employees of old departments are transferred to the new department. Heads of the old departments are compared and the more senior one is assigned as the head of the new department.
- loadDepartments: Loads the departments vector from the given path.
- increaseSalary: Increases the salary of an employee by the given increment. If an employee with given ID does not exist, an InvalidOperation exception must be generated.

Question 4: Write the source code of the method listStaffRecruitedAfter of class Company. This method lists the employees in all departments who are recruited after the given date. All departments must be searched in parallel, therefore you need to write the source code of an additional class to implement multithreading.

BLM2562 OBJECT ORIENTED PROG., Section 2, 2nd MIDTERM **(11/05/2015)**

Question 1: Write the source code of the toString method of class Employee (10p).

```
	public String toString( ) {
		String intro = "(id: " + id + ")\t" + name;
		if( worksIn != null ) {
			intro += "\t"+worksIn.getName();
			if( worksIn.getDepartmentHead() == this )
				intro += "(head)";
		}
		if( recruited != null )
			intro += "\t"+recruited;
		return intro;
	}
```

Question 2: Write the source code of class Department (35p).

```
import java.util.*;
public class Department implements java.io.Serializable {
	private static final long serialVersionUID = 1L;
	private final Company company;
	private final int id;
	private String name;
	private Hashtable<Integer,Employee> staff;
	private Employee departmentHead;

	public Department(Company company, String name, int id) {
		this.company = company; this.name = name; this.id = id;
		staff = new Hashtable<Integer,Employee>();
	}
	public Company getCompany() { return company; }
	public String getName() { return name; }
	public int getId() { return id; }
	public Employee searchEmployee( int empID ) {
		return staff.get(empID);
	}
	public Employee searchEmployee( String empName ) {
		for( Employee emp : staff.values() )
			if( emp.getName() == empName )
				return emp;
		return null;
	}
	public void addEmployee( Employee newEmp ) throws InvalidOperation {
		if( company.searchEmployeeInAllDepartments(newEmp.getId()) != null )
			throw new InvalidOperation(newEmp + " already enlisted");
		staff.put(newEmp.getId(), newEmp);
	}
	public void setDepartmentHead( int empID ) throws InvalidOperation {
		if( searchEmployee(empID) != null )
			departmentHead = searchEmployee(empID);
		throw new InvalidOperation("Department " + name +
				"'s head must be one of its employees");
	}
	public Employee getDepartmentHead() {
		return departmentHead;
	}
	public Vector<Employee> getStaff( ) {
		Vector<Employee> employees = new Vector<Employee>();
		for( Employee emp : staff.values() )
			employees.add(emp);
		return employees;
	}
}
```

Question 3: Write the source code of only the following methods of class Company (35p).

```
public void mergeDepartments( Department old1, Department old2,
                Department newDept ) throws InvalidOperation {
        departments.remove(old1);
        departments.remove(old2);
        for( Employee emp : old1.getStaff() )
                newDept.addEmployee(emp);
        for( Employee emp : old2.getStaff() )
                newDept.addEmployee(emp);
        Employee head1 = old1.getDepartmentHead();
        Employee head2 = old2.getDepartmentHead();
        if(head1.getRecruited().before(head2.getRecruited()))
                newDept.setDepartmentHead(head1.getId());
        else
                newDept.setDepartmentHead(head2.getId());
        addDepartment(newDept);
}
public void loadDepartments( String fullPath ) {
        try {
                ObjectInputStream stream = new ObjectInputStream(
                                new FileInputStream(fullPath));
                departments = (Vector<Department>)stream.readObject();
                stream.close();
        }
        catch (IOException e) {
                e.printStackTrace();
        }
        catch (ClassNotFoundException e) {
                e.printStackTrace();
        }
}
public void increaseSalary(int empID, double increment) throws InvalidOperation {
        Employee emp = searchEmployeeInAllDepartments(empID);
        if( emp != null )
                emp.setSalary(emp.getSalary()+increment);
        else throw new  InvalidOperation("Cannot find employee with id "
                        + empID + " to increase his/her salary.");
}
```

Question 4: Write the source code of the method listStaffRecruitedAfter of class Company (20p).

```
public void listStaffRecruitedAfter( Date date ) {
        for( Department dept : departments ) {
                EmployeeFinderAfterDate lister = new EmployeeFinderAfterDate(dept,date);
                Thread t = new Thread(lister);
                t.start();
        }
}

import java.util.*;
public class EmployeeFinderAfterDate implements Runnable {
        private Department dept;
        private Date date;

        public EmployeeFinderAfterDate(Department dept, Date date) {
                this.dept = dept; this.date = date;
        }
        public void run() {
                for( Employee emp : dept.getStaff() )
                        if( emp.getRecruited().after(date) )
                                System.out.println(emp);
        }
}
```