# OPT2 – portfolio opdracht

Het doel van deze portfolio opdracht is om aan te tonen dat je de lesstof (die tot nu toe behandeld is) voldoende beheerst.

Voor deze portfolio opdracht is het de bedoeling dat je een probleemstelling formuleert (max. 200 woorden) en daarbij een UML ontwerp opstelt en deze vervolgens omzet in Java met zinvolle JUnit testen.

**Beoordelingscriteria en instructies**

1. De (gehele) portfolio opdracht wordt met een voldoende afgesloten bij 55 of meer punten. Er wordt geen goed toegekend (het is of voldoende of onvoldoende).

* Aantal behaalde punten >= 55 = voldoende
* Aantal behaalde punten < 55 = onvoldoende

1. Het onderdeel wordt in zijn geheel als goed of fout beoordeeld. Het is dus NIET mogelijk een deel van de punten te krijgen voor een onderdeel/ criteria.
2. De student laat zijn werk controleren tijdens de individuele assessment.
3. De student mag voorbeelden NIET overnemen van het lesmateriaal (dit wordt gezien als fraude).
4. Als code wel werkt maar op essentiële punten onder de maat is (ontbreken van constanten, gebruiken van onjuiste variabele- en methodenamen), worden geen punten toegekend. De examinator maakt dan kenbaar waarom het onvoldoende is.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria #** | **Criteria / indicator behaald als** | **Punten** | **Onvoldoende / Voldoende** |
|  | Je een beknopte probleemstelling (max. 200 woorden) hebt geformuleerd. | 5 | Voldoende |
|  | Je een UML klassendiagram hebt bijgevoegd met minimaal 5 klassen. | 25 | Voldoende |
|  | Je overerving en polymorphism hebt toegepast. | 10 | Voldoende |
|  | Je gebruik gemaakt hebt van abstract classes en/of interfaces. | 10 | Voldoende |
|  | Je method overriding hebt toegepast. | 10 | Voldoende |
|  | Je minimaal 3 zinvolle JUnit testen hebt. | 20 | Voldoende |
|  | Je aan kunt tonen in IntelliJ dat het programma werkt (zowel het programma als de JUnit testen). | 5 | Voldoende |
|  | Je composiet en/of aggregaat relatie hebt verwerkt in de opdracht. | 15 | Voldoende |
| **Totaal** | | **100** | **100** |

**Probleemstelling**

Voor een multinational word er momenteel tijdens het sollicatie proces gebruik gemaakt van een handmatige filtersysteem. Alle sollicitanten sturen hun CV op via de e-mail waarbij de medewerkers van Human Resources handmatig bekijken of het relevant is voor hun toegewezen afdeling of niet.

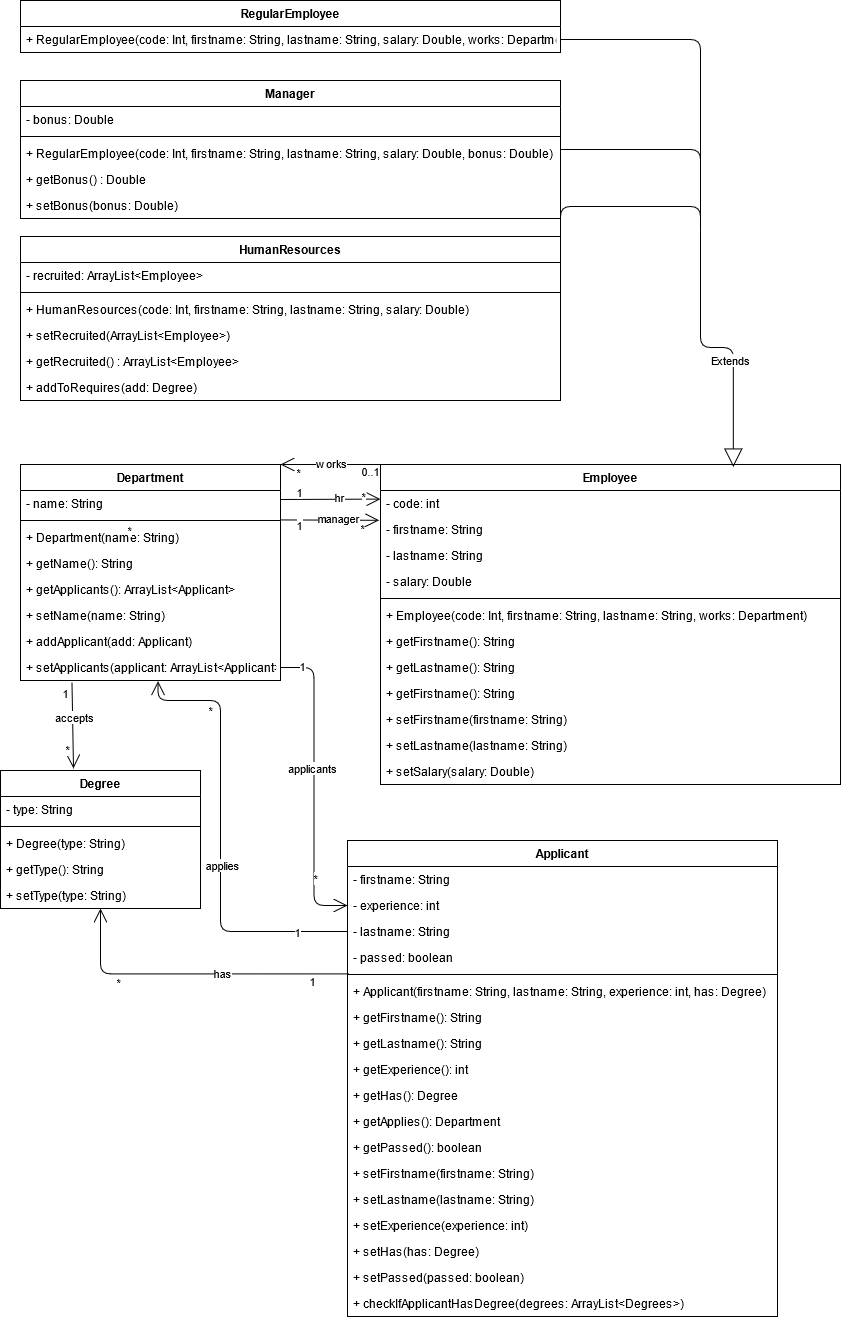
De medewerkers van Human Resources hebben elk een afdeling toegewezen waar ze verantwoordelijk voor zijn met elk afdeling die bepaalde criteria heeft voor de medewerkers. De administratie afdeling eist dat sollicitanten een mbo niveau 4 diploma hebben omtrent administratie terwijl de ICT afdeling eist dat de sollicitanten minimaal beschikken over een HBO diploma.

Alle sollicitanten worden momenteel handmatig bekeken, en als ze niet aan de eisen voldoen, worden ze aan de kant gezet voor een kandidaat die wel over die eisen beschikt. Daardoor moet het nieuwe systeem dit meer geautomatiseerd doen zodat alleen de juiste kandidaten tevoorschijn komen.

Ook moet er een geautomatiseerde manier komen om alle medewerkers te laten zien van een afdeling, aangezien het nu handmatig gedaan word. Ook het mogelijkheid om te kunnen bekijken welke medewerkers een medewerker van HR heeft gerekruteerd

**Zie volgende pagina voor UML**

**UML**



**Zie volgende pagina voor JAVA code**

**Java code**

import org.junit.Assert;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 // Human Resources  
 HumanResources hr = new HumanResources(120,"Human", "Resources", 250.000);  
  
 // Diploma's ICT  
 Degree ictDiploma = new Degree("HBO ICT");  
 Degree niv4Diploma = new Degree("Niv 4 applicatie en media ontwikkelaar");  
  
 // Manager ICT  
 Manager ahmed = new Manager(252, "Ahmed", "Deedad", 300.000, 75.000);  
  
 // Afdeling ICT  
 Department ict = new Department("ICT", ahmed, hr);  
  
 // Voeg diploma's toe aan ICT  
 ict.addToRequires(ictDiploma);  
 ict.addToRequires(niv4Diploma);  
  
 // Voeg medewerkers toe aan ICT  
 RegularEmployee kees = new RegularEmployee(254, "Kees", "Jan", 100.000, ict);  
 RegularEmployee mahmud = new RegularEmployee(255, "Mahmud", "Test", 120.000, ict);  
  
 // Voegt medewerkers toe aan de Human resources  
 hr.addToRecruited(ahmed);  
 hr.addToRecruited(kees);  
  
 // Random diploma's die niet overeenkomen  
 Degree random1 = new Degree("Zorg");  
 Degree random2 = new Degree("Administratie");  
  
  
 // Testdata andere applicants  
 ict.addToApplicants(new Applicant("Applicant", "One", 2, ictDiploma, ict));  
 ict.addToApplicants(new Applicant("Two", "Applicant", 5, ictDiploma, ict));  
  
 // Sollicatie proces  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("-------------------------------------------------------------------");  
 System.*out*.println("Voornaam:");  
 System.*out*.println("-------------------------------------------------------------------");  
 String firstname = scanner.nextLine();  
 System.*out*.println("-------------------------------------------------------------------");  
 System.*out*.println("Achternaam:");  
 System.*out*.println("-------------------------------------------------------------------");  
 String lastname = scanner.nextLine();  
 System.*out*.println("-------------------------------------------------------------------");  
 System.*out*.println("Werkervaring:");  
 System.*out*.println("-------------------------------------------------------------------");  
 int experience = scanner.nextInt();  
 scanner.nextLine();  
 System.*out*.println("-------------------------------------------------------------------");  
 System.*out*.println("Welk diploma heb je? Kies uit de volgende (typ het precies na):");  
 System.*out*.println("-------------------------------------------------------------------");  
 System.*out*.println(ictDiploma.getType());  
 System.*out*.println(niv4Diploma.getType());  
 System.*out*.println(random1.getType());  
 System.*out*.println(random2.getType());  
 System.*out*.println("-------------------------------------------------------------------");  
 String degree = scanner.nextLine();  
 System.*out*.println("-------------------------------------------------------------------");  
  
 // Kijk of hij de benodigde diploma heb, zo wel, maak een nieuwe applicant aan als het zo is, anders negeren  
 for (Degree requiredDegrees : ict.getRequires()) {  
 if (requiredDegrees.getType().equals(degree)) {  
 Applicant applicant = new Applicant(firstname, lastname, experience, requiredDegrees, ict);  
 ict.addToApplicants(applicant);  
 }  
 }  
  
 // Laat zien wie Human Resources allemaal heeft gerekruteerd  
 for (Employee recruited : hr.getRecruited()) {  
 System.*out*.println(recruited.getFirstname() + recruited.getLastname());  
 }  
  
 // Laat zien wie allemaal heeft gesoliciteerd bij de ICT afdeling  
 for (Applicant x: ict.getApplicants()) {  
 System.*out*.println("Voornaam:" + x.getFirstname() + " Achternaam:" + x.getLastname() + " Werkervaring:" + x.getExperience() + " Diploma:" + x.getHas().getType());  
 }  
 }  
}  
  
abstract class Employee {  
 private int code;  
 private String firstname;  
 private String lastname;  
 private Double salary;  
 private Department works;  
  
 public Employee(int code, String firstname, String lastname, Double salary) {  
 this.code = code;  
 this.firstname = firstname;  
 this.lastname = lastname;  
 this.salary = salary;  
 }  
  
 public Employee(int code, String firstname, String lastname, Double salary, Department works) {  
 this.code = code;  
 this.firstname = firstname;  
 this.lastname = lastname;  
 this.salary = salary;  
 this.works = works;  
 }  
  
 public int getCode() {  
 return code;  
 }  
  
 public void setCode(int code) {  
 this.code = code;  
 }  
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public void setFirstname(String firstname) {  
 this.firstname = firstname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public Double getSalary() {  
 return salary;  
 }  
  
 public void setSalary(Double salary) {  
 this.salary = salary;  
 }  
  
 public Department getWorks() {  
 return works;  
 }  
  
 public void setWorks(Department works) {  
 this.works = works;  
 }  
}  
  
class Department {  
 private String name;  
 private Employee manager;  
 private Employee hr;  
 private ArrayList<Degree> requires = new ArrayList<Degree>();  
 private ArrayList<Applicant> applicants = new ArrayList<Applicant>();  
  
 public Department(String name, Employee manager, Employee hr){  
 this.name = name;  
 this.manager = manager;  
 this.hr = hr;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Employee getManager() {  
 return manager;  
 }  
  
 public void setManager(Employee manager) {  
 this.manager = manager;  
 }  
  
 public Employee getHr() {  
 return hr;  
 }  
  
 public void setHr(Employee hr) {  
 this.hr = hr;  
 }  
  
 public ArrayList<Applicant> getApplicants() {  
 return applicants;  
 }  
  
 public void setApplicants(ArrayList<Applicant> applicants) {  
 this.applicants = applicants;  
 }  
  
 public ArrayList<Degree> getRequires() {  
 return requires;  
 }  
  
 public void setRequires(ArrayList<Degree> requires) {  
 this.requires = requires;  
 }  
  
 public void addToRequires(Degree add) {  
 boolean addToRequires = true;  
  
 for (Degree recruited: this.getRequires()) {  
 if (recruited.getType().equals(add.getType())) {  
 addToRequires = false;  
 }  
 }  
  
 if(addToRequires){  
 this.requires.add(add);  
 }  
 }  
  
 public boolean addToApplicants(Applicant add) {  
 boolean addToApplicants = true;  
  
 for (Applicant applicant: this.getApplicants()) {  
 String name = applicant.getFirstname()+applicant.getLastname();  
 String nameNew = add.getFirstname()+add.getLastname();  
 if (name.equals(nameNew)) {  
 addToApplicants = false;  
 }  
 }  
  
 if(addToApplicants){  
 this.applicants.add(add);  
 }  
  
 return addToApplicants;  
 }  
}  
  
class RegularEmployee extends Employee {  
  
 public RegularEmployee(int code, String firstname, String lastname, Double salary) {  
 super(code, firstname, lastname, salary);  
 }  
  
 public RegularEmployee(int code, String firstname, String lastname, Double salary, Department works) {  
 super(code, firstname, lastname, salary, works);  
 }  
  
 @Override  
 public int getCode() {  
 return 997 + super.getCode();  
 }  
}  
  
class Manager extends Employee {  
 private Double bonus;  
  
 public Manager(int code, String firstname, String lastname, Double salary, Double bonus) {  
 super(code, firstname, lastname, salary);  
 this.bonus = bonus;  
 }  
  
 @Override  
 public int getCode() {  
 return 998 + super.getCode();  
 }  
  
 public Double getBonus() {  
 return bonus;  
 }  
  
 public void setBonus(Double bonus) {  
 this.bonus = bonus;  
 }  
}  
  
class HumanResources extends Employee {  
 private ArrayList<Employee> recruited = new ArrayList<Employee>();  
  
 public HumanResources(int code, String firstname, String lastname, Double salary) {  
 super(code, firstname, lastname, salary);  
 }  
  
 @Override  
 public int getCode() {  
 return 999 + super.getCode();  
 }  
  
 public ArrayList<Employee> getRecruited() {  
 return recruited;  
 }  
  
 public void setRecruited(ArrayList<Employee> recruited) {  
 this.recruited = recruited;  
 }  
  
 public void addToRecruited(Employee add) {  
 boolean addToRecruited = true;  
  
 for (Employee recruited: this.getRecruited()) {  
 if (recruited.getCode() == add.getCode()) {  
 addToRecruited = false;  
 }  
 }  
  
 if(addToRecruited){  
 this.recruited.add(add);  
 }  
 }  
}  
  
class Degree {  
 private String type;  
  
 public Degree(String type) {  
 this.type = type;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public void setType(String type) {  
 this.type = type;  
 }  
}

class Applicant {  
 private String firstname;  
 private String lastname;  
 private int experience;  
 private Degree has;  
 private Department applies;  
 private boolean passed;  
  
 public Applicant(String firstname, String lastname, int experience, Degree has, Department applies) {  
 this.firstname = firstname;  
 this.lastname = lastname;  
 this.experience = experience;  
 this.has = has;  
 this.applies = applies;  
 }  
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public void setFirstname(String firstname) {  
 this.firstname = firstname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public int getExperience() {  
 return experience;  
 }  
  
 public void setExperience(int experience) {  
 this.experience = experience;  
 }  
  
 public Degree getHas() {  
 return has;  
 }  
  
 public void setHas(Degree has) {  
 this.has = has;  
 }  
  
 public Department getApplies() {  
 return applies;  
 }  
  
 public void setApplies(Department applies) {  
 this.applies = applies;  
 }  
  
 public boolean isPassed() {  
 return passed;  
 }  
  
 public void setPassed(boolean passed) {  
 this.passed = passed;  
 }  
  
 public boolean checkIfApplicantHasDegree(ArrayList<Degree> degrees) {  
 boolean has = false;  
  
 for (Degree degree: degrees) {  
 if (degree.getType().equals(this.getHas().getType())) {  
 has = true;  
 }  
 }  
  
 return has;  
 }  
}

**Junit**

import org.junit.Assert;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class DepartmentTest {  
  
 @Test  
 void addToApplicantsNonExisting() {  
 // Human Resources  
 HumanResources hr = new HumanResources(120,"Human", "Resources", 250.000);  
  
 // Diploma's ICT  
 Degree ictDiploma = new Degree("HBO ICT");  
 Degree niv4Diploma = new Degree("Niv 4 applicatie en media ontwikkelaar");  
  
 // Manager ICT  
 Manager ahmed = new Manager(252, "Ahmed", "Deedad", 300.000, 75.000);  
  
 // Afdeling ICT  
 Department ict = new Department("ICT", ahmed, hr);  
  
 Assert.*assertTrue*(ict.addToApplicants(new Applicant("Applicant", "One", 2, ictDiploma, ict)));  
 }  
  
 @Test  
 void addToApplicantsExisting() {  
 // Human Resources  
 HumanResources hr = new HumanResources(120,"Human", "Resources", 250.000);  
  
 // Diploma's ICT  
 Degree ictDiploma = new Degree("HBO ICT");  
 Degree niv4Diploma = new Degree("Niv 4 applicatie en media ontwikkelaar");  
  
 // Manager ICT  
 Manager ahmed = new Manager(252, "Ahmed", "Deedad", 300.000, 75.000);  
  
 // Afdeling ICT  
 Department ict = new Department("ICT", ahmed, hr);  
  
 // Voeg applicant toe  
 ict.addToApplicants(new Applicant("Applicant", "One", 2, ictDiploma, ict));  
  
 Assert.*assertFalse*(ict.addToApplicants(new Applicant("Applicant", "One", 2, ictDiploma, ict)));  
 }  
  
 @Test  
 void checkIfHeHasTheDegreeWithTrue() {  
 // Human Resources  
 HumanResources hr = new HumanResources(120, "Human", "Resources", 250.000);  
  
 // Diploma's ICT  
 Degree ictDiploma = new Degree("HBO ICT");  
 Degree niv4Diploma = new Degree("Niv 4 applicatie en media ontwikkelaar");  
 Degree random1 = new Degree("Zorg");  
  
 // Manager ICT  
 Manager ahmed = new Manager(252, "Ahmed", "Deedad", 300.000, 75.000);  
  
 // Afdeling ICT  
 Department ict = new Department("ICT", ahmed, hr);  
 ict.addToRequires(ictDiploma);  
 ict.addToRequires(niv4Diploma);  
  
 // Applicant  
 Applicant applicant = new Applicant("Applicant", "One", 2, ictDiploma, ict);  
  
 Assert.*assertTrue*(applicant.checkIfApplicantHasDegree(ict.getRequires()));  
 }  
  
 @Test  
 void checkIfHeHasTheDegreeWithFalse() {  
 // Human Resources  
 HumanResources hr = new HumanResources(120, "Human", "Resources", 250.000);  
  
 // Diploma's ICT  
 Degree ictDiploma = new Degree("HBO ICT");  
 Degree niv4Diploma = new Degree("Niv 4 applicatie en media ontwikkelaar");  
 Degree random1 = new Degree("Zorg");  
  
 // Manager ICT  
 Manager ahmed = new Manager(252, "Ahmed", "Deedad", 300.000, 75.000);  
  
 // Afdeling ICT  
 Department ict = new Department("ICT", ahmed, hr);  
 ict.addToRequires(ictDiploma);  
 ict.addToRequires(niv4Diploma);  
  
 // Applicant  
 Applicant applicant = new Applicant("Applicant", "One", 2, random1, ict);  
  
 Assert.*assertFalse*(applicant.checkIfApplicantHasDegree(ict.getRequires()));  
 }  
}