**public** **abstract** **class** Researcher { ……………../ 1 ……………../ 3

**private** String name;

**public** Researcher(String s) { ……………../ 1

name = s;

}

**public** **abstract** **double** getSalary(); ……………../ 1

}

**public** **class** Faculty **extends** Researcher { ……………../ 1 ……………../ 4

**private** String position;

**private** **int** nbYearsOfExperience;

**public** Faculty(String name, String p, **int** nbE) {

**super**(name); ……………../ 1

position = p;

nbYearsOfExperience = nbE;

}

**public** **double** getSalary() { ……………../ 1

**return** 5600 + (500 \* nbYearsOfExperience);

}

**public** **int** getNbYearsOfExperience() { ……………../ 1

**return** nbYearsOfExperience;

}

}

**public** **class** Project { ……………../ 34

**private** String name;

**private** **int** duration;

**private** **double** budget;

**private** Researcher arRes[]; ……………../ 1

**private** **int** nbRes; ……………../ 1

**public** Project(String s, **int** d, **double** b,**int** size) {

name = s;

duration = d;

budget = b;

arRes = **new** Researcher[size]; ……………../ 1

nbRes = 0; ……………../ 1

}

**public** **void** addResearcher(Researcher r)

**throws** ArrayIndexOutOfBoundsException, Exception { ……/1 .../9

**if** (nbRes < arRes.length) { ……………../ 1

**if** ( r **instanceof** Student ……………../ 1

&& ((Student)r).getGpa() < 2.0) ……………../ 1 + 1

**throw** **new** Exception("Not Allowed to …"); ……/1

arRes[nbRes] = r; ……………../ 1

nbRes++; ……………../ 1

}

**else**

**throw** **new** ArrayIndexOutOfBoundsException(); ……………../ 1

}

**public** **int** addfromArray(Researcher ar[], **double** s) { ……………../ 10

**int** n = 0; ……………../ 1

**for** (**int** i = 0; i < ar.length; i++) { ……………../ 1

**if** ( ar[i] != **null** &&

ar[i] **instanceof** Faculty && ……………../ 1

((Faculty)ar[i]).getSalary() > s && ……………../ 1 + 1

nbRes < arRes.length) { ……………../ 1

arRes[nbRes] = ar[i]; ……………../ 1

nbRes ++; ……………../ 1

n++; ……………../ 1

}

}

**return** n; ……………../ 1

}

**public** String getName() { ……………../ 1

**return** name;

}

**public** **double** getBudget() { ……………../ 1

**return** budget;

}

**public** Faculty getFaculty (**double** s, **double** g) { ……………../ 9

Faculty f = **null**; ……………../ 1

**for** (**int** i = 0 ; i < nbRes; i++) { ……………../ 1

**if** ( arRes[i] **instanceof** Faculty ) { ……………../ 1

**if** ( f == **null** || ……………../ 1

f.getNbYearsOfExperience() < ……………../ 1

((Faculty)arRes[i]).getNbYearsOfExperience())…/1+1

f = (Faculty) arRes[i]; ……………../ 1 + 1

}

}

**return** f; ……………../ 1

// Other Solution

/\* int index = -1; ……………../ 0.5

double max = 0.0; ……………../ 0.5

for (int i = 0 ; i < nbRes; i++) { ……………../ 1

if ( arRes[i] instanceof Faculty ) { ……………../ 1

if ( max <

((Faculty)arRes[i]).getNbYearsOfExperience())./1+1

index = i; ……………../ 1

max =((Faculty)arRes[i]). ……………../ 1

getNbYearsOfExperience();……………../ 1

}

}

}

if (index == -1)

return null;

else

return (Faculty) arRes[index]; ……………../ 0.5 + 0.5

\*/

}

}

**public** **class** LinkedListOfProjects { ……………../ 35

**private** Node head; ……………../ 1

**public** LinkedListOfProjects() {

head = **null**; ……………../ 1

}

**public** **double** getAverageBudget() { ……………../ 8

**double** sum = 0.0; ……………../ 1

**int** n = 0; ……………../ 1

Node current = head; ……………../ 1

**while** (current != **null**) { ……………../ 1

sum += current.getData().getBudget(); ……………../ 1

n++; ……………../ 1

current = current.getNext(); ……………../ 1

}

**if** (n != 0)

**return** sum / n; ……………../ 1

**else**

**return** 0.0;

}

**public** **boolean** remove(Project p) { ……………../ 11

Node current = head; ……………../ 1

Node previous = **null**; ……………../ 1

**while** ( current != **null** && ……………../ 1

! current.getData().getName().equals(p.getName())){ …/1

previous = current; ……………../ 1

current = current.getNext();……………../ 1

}

**if** (current != **null**) { ……………../ 1

**if** (previous == **null**) ……………../ 1

head = current.getNext();……………../ 1

**else**

previous.setNext(current.getNext());……………../ 1

current.setNext(**null**);

**return** **true**; ……………../ 0.5

}

**else**

**return** **false**; ……………../ 0.5

}

**public** **int** size() { ……………../ 6

**int** n = 0; ……………../ 1

Node current = head; ……………../ 1

**while** (current != **null**) { ……………../ 1

n++; ……………../ 1

current = current.getNext(); ……………../ 1

}

**return** n; ……………../ 1

}

**public** Project get(**int** i) { ……………../ 8

Node current = head; ……………../ 1

**int** j = 0; ……………../ 1

**while** (current != **null** && j < i) { ……………../ 1

current = current.getNext(); ……………../ 1

j++; ……………../ 1

}

**if** (current != **null**) ……………../ 1

**return** current.getData(); ……………../ 1

**else**

**return** **null**; ……………../ 1

// Other Solution

/\* Node current = head; ……………../ 1

if (i < 0 || i >= size()) ……………../ 1+1

return null; ……………../ 1

else { ……………../ 1

for (int j=0; j < i; j++) ……………../ 1

current =current.getNext(); ……………../ 1

return current.getData(); ……………../ 1

}

\*/ }

}

**public** **class** College { ……………../ 24

**private** String name;

**private** LinkedListOfProjects myList; ……………../ 1

**public** College(String s) {

name = s;

myList = **new** LinkedListOfProjects(); ……………../ 1

}

**public** Project HighestBudgetProject() { ……………../ 5

Project max = myList.get(0); ……………../ 1

**for** (**int** i = 1 ; i < myList.size(); i++) { ……………../ 1

**if** (myList.get(i).getBudget() > max.getBudget()) …./ 1

max = myList.get(i); ……………../ 1

}

**return** max; ……………../ 1

}

**public** **int** saveIntoFile(String fname, **double** b)

**throws** IOException { ……………../ 7

File f = **new** File(fname); ……………../ 0.5

FileOutputStream fos = **new** FileOutputStream(f); ……………../ 0.5

ObjectOutputStream obF = **new** ObjectOutputStream(fos); ………../ 0.5

**int** n = 0; ……………../ 1

**for** (**int** i = 0; i < myList.size(); i++) { ……………../ 1

**if** (myList.get(i).getBudget() > b) { ……………../ 1

obF.writeObject(myList.get(i)); ……………../ 1

n++; ……………../ 1

}

}

obF.close();

**return** n; ……………../ 0.5

}

**public** **int** readFromFileAndRemove(String fname) **throws** IOException { /10

File f = **new** File(fname); ……………../ 0.5

FileInputStream fs = **new** FileInputStream(f); ……………../ 0.5

ObjectInputStream obF = **new** ObjectInputStream(fs); ………../ 0.5

Project p;

**int** n = 0; ……………../ 1

**try** { ……………../ 1

**while** (**true**) { ……………../ 1

p = (Project) obF.readObject(); ……………../ 1 + 1

**if** ( myList.remove(p) ) ……………../ 1

n++; ……………../ 1

}

}**catch**(EOFException e) { ……………../ 1

System.*out*.println("EOF is reached .....");

} **catch** (Exception e) {

e.printStackTrace();

}

obF.close();

**return** n; ……………../ 0.5

}

}