**Exercise1:**



***Award*** class***:***

* + Attributes:
		- ***name***: the name of the award.
		- ***awardYear:*** the year of the award.
		- ***cash:*** the amount of money obtained if the award is won.
	+ Methods:
* ***Award (name: String, year: int, cash: double)***: constructor
* ***getYear(*):** this method returns the year of the award. Public
* ***getCash(*):** this method returns the cash of the award.

***Role*** class

* + Attributes:
		- ***roleName***: the name of the role.
		- ***nbOccur:*** the number of occurrences of the role in the movie.
		- ***totalTime:*** the total time allocated for the role in the movie.
	+ Methods:
* ***Role (name: String, nbOccur: int, totalTime: int)***: constructor.
* ***getNbOccur(*):** this method returns the number of occurrences of the role.
* ***calculateCachet(*):**this method calculates and returns the cachet (the salary) earned by the actor playing the role. The cachet is calculated as follows:
	+ ***PrincipalRole***:

*cachet = basic salary + number of occurrences of the role \* 2000.*

* + ***SecondaryRole***:

*cachet = hourly rate \* total time allocated for the role.*

* ***calculateTotalRevenue ():***this method returns the total cash of the *Awards* won by the role. If the role does not have any award, this method throws an exception.

***PrincipalRole*** class:

* + Attributes:
		- ***basicSalary***: the basic salary allocated for the role.
	+ Methods:
* ***PrincipalRole (name: String, nbOccur: int, totalTime: int, basicSalary: double):*** constructor.

***SecondaryRole*** class:

* + Attributes:
		- ***hourlyRate***: the hourly rate allocated for the role.
	+ Methods:
* ***SecondaryRole (name: String, nbOccur: int, totalTime: int, hourlyRate: double):*** constructor.

**QUESTION**: Translate into Java code the following classes:

* ***Role*** *(do not implement the* ***getNbOccur*** *method)****,***
* ***PrincipalRole.***

**Exercise 2:**

Let’s consider the same class ***Role*** described in exercise 1.



***Movie class***

* + Attributes:
		- ***name***: the name of the movie.
		- ***year:*** the year of the movie.
	+ Methods:
* ***Movie (name: String, year: int, size: int)***: constructor.
* ***addRole(r: Role*):**this method adds the *Role* ***r*** to the Movie. It returns *true* if the *Role* ***r*** is successfully added. Otherwise, it returns *false*.
* ***countPrincipalRoles(nb: int):***this method returns the number of principal roles of the movie having the number of occurrences greater than ***nb***.
* ***getTotalCachetOfSecondaryRoles(hr: double):*** this method returns the total cachet of secondary roles having a hourly rate equal to hr. If the parameter hr is negative this method throws an exception with the following message “*Negative Argument*”.
* ***getPrincipalRoles():***this method returns an array containing every principal role object which its total revenue is greater than its cachet .
* ***splitAndSave(pActFilename: String, sActFileName: String, c: double):***this method saves the *PrincipalRole* objects having a cachet greater than ***c*** in the file *pActFileName*. Also, it saves the the *SecondaryRole* objects having a cachet less than ***c*** in the file *sActFileName*.

**QUESTION**: Translate into Java code the class ***Movie.***

**public** **abstract** **class** Role {

 **private** String roleName;

 **protected** **int** nbOccur;

 **protected** **int** totalTime;

 **private** Award arrAwards[];

 **private** **int** nbA;

 **public** Role(String name, **int** nbOccur, **int** totalTime)

 {

 roleName = name;

 **this**.nbOccur = nbOccur;

 **this**.totalTime = totalTime;

 arrAwards = **new** Award[10];

 nbA = 0;

 }

 **public** **int** getNBOccur()

 {

 **return** nbOccur;

 }

 **public** **abstract** **double** calculateCachet();

 **public** **double** calculateTotalRevenue() **throws** Exception

 {

 **if**(nbA == 0)

 **throw** **new** Exception("No Awards");

 **double** sum=0;

 **for**(**int** i=0; i < nbA; i++)

 {

 sum+=arrAwards[i].getCash();

 }

 **return** sum;

 }

 **public** Role(Role r)

 {

 roleName = r.roleName;

 nbOccur = r.nbOccur;

 totalTime = r.totalTime;

arrAwards = **new** Award[10];

 **for**(**int** i=0; i<r.nbA; i++)

 {

 arrAwards[i] = r.arrAwards[i];

 }

 nbA = r.nbA;

 }

}

**public** **class** PrincipalRole **extends** Role{

 **private** **double** basicSalary;

 **public** PrincipalRole(String name, **int** nbOccur, **int** totalTime, **double** basicSalary) {

 **super**(name, nbOccur, totalTime);

 **this**.basicSalary = basicSalary;

 }

 **public** **double** calculateCachet() {

 **return** basicSalary + getNBOccur() \* 2000;

 }

 **public** PrincipalRole(PrincipalRole p)

 {

 **super**(p);

 basicSalary = p.basicSalary;

 }

}

**public** **class** Movie {

 **private** String name;

 **private** **int** year;

 **private** Role[] arrRoles;

 **private** **int** nbR;

 **public** Movie(String name, **int** year, **int** size)

 {

 **this**.name = name;

 **this**.year = year;

 arrRoles = **new** Role[size];

 nbR = 0;

 }

 **public** **boolean** addRole(Role r)

 {

 **if**(nbR >= arrRoles.length)

 **return** **false**;

 **if**(r **instanceof** SecondaryRole)

 arrRoles[nbR] = **new** SecondaryRole( (SecondaryRole) r);

 **else**

 arrRoles[nbR] = **new** PrincipalRole( (PrincipalRole) r);

 nbR++;

 **return** **true**;

 }

 **public** **int** countPrincipalRoles(**int** nb)

 {

 **int** counter = 0;

 **for**(**int** i=0; i<nbR; i++)

 {

 **if**(arrRoles[i] **instanceof** PrincipalRole)

 **if**(arrRoles[i].getNBOccur() > nb)

 counter++;

 }

 **return** counter;

 }

 **public** **double** getTotalCachetOfSecondaryRoles(**double** hr) **throws** Exception

 {

 **if**(hr < 0)

 **throw** **new** Exception("Negative Argument");

 **double** sum = 0;

 **for**(**int** i=0; i<nbR; i++)

 {

 **if**(arrRoles[i] **instanceof** SecondaryRole)

 {

 SecondaryRole r = (SecondaryRole)arrRoles[i];

 **if**(r.getHourlyRate() == hr)

 {

 sum += r.calculateCachet();

 }

 }

 }

 **return** sum;

 }

 **public** PrincipalRole[] getPrincipalRoles()

 {

 PrincipalRole[] r = **new** PrincipalRole[nbR];

 **int** j=0;

 **for**(**int** i=0; i<nbR; i++)

 {

 **if**(arrRoles[i] **instanceof** PrincipalRole)

 **try** {

 **if**(arrRoles[i].calculateTotalRevenue() > arrRoles[i].calculateCachet())

 r[j++] = (PrincipalRole) arrRoles[i];

 }

 **catch** (Exception e) {

 System.*out*.print(e.getMessage());

 }

 }

 **return** r;

 }

}