Object-oriented programming – Java

Inheritance Pt1

mgr Sara Jurczyk

Exercise 1 A.

`BlueCollarWorker`	`WhiteCollarWorker`
• name • surname • PESEL • Year of birth • Year of employment • Gender • `company` • `occupation` • payment per 1 working hour • number of working hours	• name • surname • PESEL • Year of birth • Year of employment • Gender • `company` • `occupation` • `salary` (monthly salary)
• non-parametric constructor • consructor setting all fields based on parameters • methods `get` • methods `set` • method `toString` • method `getAge`, returning the age of employee • method `calculateJobSeniority`, that calculates and returns his job seniority • method `calculatePayment`, that returns total payment • method `calculateTimeToRetirement`, that calculates and returns the number of years to retirement (for women it is 60 years old, for men 65 years) • method `display`, that displays all information about the employee	• non-parametric constructor • consructor setting all fields based on parameters • methods `get` • methods `set` • method `toString` • method `getAge`, returning the age of employee • method `calculateJobSeniority`, that calculates and returns his job seniority • method `calculateTimeToRetirement`, that calculates and returns the number of years to retirement (for women it is 60 years old, for men 65 years) • method `display`, that displays all information about the employee

(hint: to get the current year use `Calendar.getInstance().get(Calendar.YEAR)`)

Zadanie 1 B.

Extend the project from the exercise 1A and add class `CommissionWorker`. Information about monthly incomes are stored in an array `monthlySalaries`. Besides attributes and methods characteristic for all employees, the class has additional methods:

- `calculateAverageSalary`, that returns the average salary calculated for all incomes in the array
- `calculateTotalIncome`, that returns the total sum of incomes in the array
- `calculateBiggerSalaries`, that returns the number od elements/incoms in the array bigger that the floating-point number passed as the parameter

Exercise 2.

Write a declaration of a **Person** class with the following protected fields:

- *name*
- *surname*
- *age*
- *nationality*

The class contains the following public methods:

- a non-parametric constructor (does nothing)
- constructor setting all fields based on its parameters
- methods *get* that return values of the class fields
- methods *set* that set values of the class fields based on its parameters

Then, write a declaration of a **Student** class that inherits from the Person class. The Student class has the following private fields:

- *field_of_study*
- *semester* // numbers from 1 to 6

The class contains the following public methods:

- constructor setting all fields based on its parameters
- methods *get* that return values of the class fields
- method *toString* that displays all information about the student
- method *nextSemester* that increases *semester* by 1 or displays "You have graduated!".