Java → Object-oriented programming → Inheritance and polymorphism → The keyword super

4663 users solved this problem. Latest completion was about 14 hours ago.

The keyword super → Employees

Medium 14 minutes

Develop a class hierarchy of employees. The hierarchy should include three classes:

- **Employee** is the base class. It includes three fields (`name`, `email` and integer variable `experience`), one constructor with three arguments and three getters: `getName()`, `getEmail()`, `getExperience()`.
- **Developer** is a subclass. It includes fields from the base class and two additional fields (`mainLanguage`, `skills`), one constructor with five arguments and two getters: `getMainLanguage()`, `getSkills()`.
- **DataAnalyst** is another subclass. It includes fields from the base class and two additional fields (`phd`, `methods`), one constructor with five arguments and two getters: `isPhd()`, `getMethods()`.

You need **to define types of the fields and write suitable constructors.** To understand it see the code below.

```
String[] skills = { "git", "Scala", "JBoss", "UML" };
Developer developer = new Developer("Mary", "mary@mail.com", 3, "Java", skills);

String[] methods= { "neural networks", "decision tree", "bayesian algorithms" };
DataAnalyst analyst = new DataAnalyst("John", "john@gmail.com", 2, true, methods);
```

It should work correctly with your class hierarchy. Do not forget to write getters with the specified name (otherwise the test system won't be able to check your solution).

Do not make your classes `public`!

Report a typo

Write a program

Code Editor IDE

Java

```
class Employee {

    // write fields
    protected String name;
    protected String email;
    protected int experience;

    // write constructor
    public Employee(String name, String email, int experience) {
        this.name = name;
        this.email = email;
        this.experience = experience;
    }

    // write getters
    protected String getName() {
        return name;
    }

    protected String getEmail() {
        return email;
    }

    protected int getExperience() {
        return experience;
    }
}

class Developer extends Employee {

    // write fields
    protected String mainLanguage;
    protected String[] skills;

    // write constructor
    public Developer(String name, String email, int experience, String mainLanguage, String[] skills) {
        super(name, email, experience);
        this.mainLanguage = mainLanguage;
        this.skills = skills.clone();
    }

    // write getters
    protected String getMainLanguage() {
        return mainLanguage;
    }

    protected String[] getSkills() {
        return skills.clone();
    }
}

class DataAnalyst extends Employee {

    // write fields
    public boolean phd;
    public String[] methods;

    // write constructor
    public DataAnalyst(String name, String email, int experience, boolean phd, String[] methods) {
        super(name, email, experience);
        this.phd = phd;
        this.methods = methods.clone();
    }

    // write getters
    protected boolean isPhd() {
        return phd;
    }

    protected String[] getMethods() {
        return methods.clone();
    }
}

```

✓ Correct.

That's an awesome solution! What do you think about showing it off? Post it to Solutions so other learners can enjoy it too.

391 users liked this problem. **52** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Continue Solve again Solutions (87)

Time limit: 8 seconds Memory limit: 256 MB

Comments (58) Hints (9) Useful links (1) Solutions (87) Show discussion