Lab: Inheritance

Problems for exercises and homework for the "Java OOP Basics" course @ SoftUni.

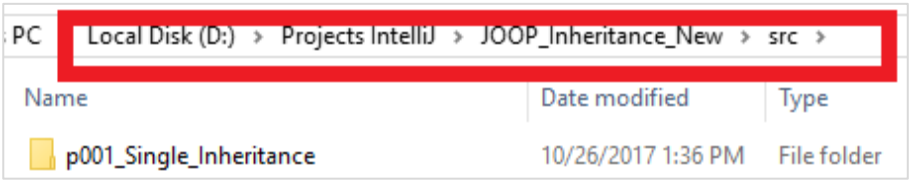
You can check your solutions here: https://judge.softuni.bg/Contests/478/Inheritance-Lab .

Part I: Upload Solutions

In that exercise Judge uses Reflection to check your solutions.

If you are using **packages for every problem** you should:

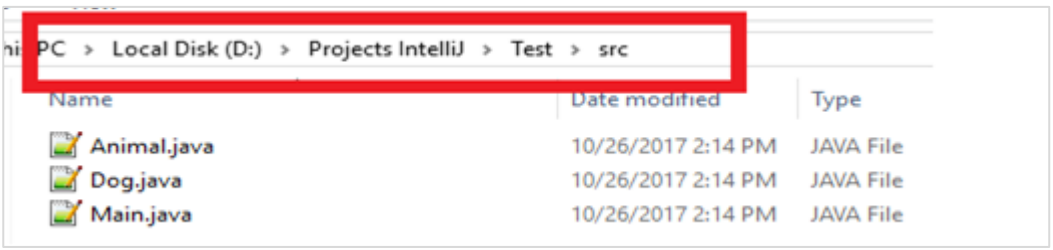
- Go to project folder
- Go to src folder



- Send the **whole package** to zip and upload.

If you are **creating project for every problem**, you should:

- Go to project folder
- Go to src folder:



- Send **everything** to zip and upload.

Part II: Inheritance

1. Single Inheritance

Create two classes named **`Animal`** and **Dog**.

`Animal` with a single public method **`eat()`** that prints: **"eating…"**

Dog with a single public method **`bark()`** that prints: **"barking…"**

Dog should inherit from **`Animal`**.

```
public static void main(String[] args) {

    Dog dog = new Dog();
    dog.eat();
    dog.bark();
}
```

```
Run  Main (2)
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ...
eating...
barking...
```

Hints

Use the **extends** keyword to build a hierarchy

2. Multiple Inheritance

Create three classes named **`Animal`**, **Dog** and **Puppy**.

`Animal` with a single public method **`eat()`** that prints: **"eating…"**

Dog with a single public method **`bark()`** that prints: **"barking…"**

Puppy with a single public method **`weep()`** that prints: **"weeping…"**

Dog should inherit from **`Animal`**. **Puppy** should inherit from **Dog**.

```
Puppy puppy = new Puppy();
puppy.eat();
puppy.bark();
puppy.weep();
```

```
Run  Main (4)
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ..
eating...
barking...
weeping...
```

3. Hierarchical Inheritance

Create three classes named **`Animal`**, **Dog** and **`Cat`**.

`Animal` with a single public method **`eat()`** that prints: **"eating…"**

Dog with a single public method **`bark()`** that prints: **"barking…"**

`Cat` with a single public method **`meow()`** that prints: **"meowing…"**

Dog and **`Cat`** should inherit from **`Animal`**.

```java
public static void main(String[] args) {

    Dog dog = new Dog();
    dog.eat();
    dog.bark();

    Cat cat = new Cat();
    cat.eat();
    cat.meow();

}
```

```
Run  Main (3)
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ...
eating...
barking...
eating...
meowing...
```

Part III: Reusing Classes

4. Fragile Base Class

Create three classes named **`Animal`**, **`Predator`** and **`Food`**.

`Predator` should inherit from **`Animal`**.

`Animal:`

- Protected field: **`foodEaten: ArrayList<Food>`**
- Public final method: **`eat(Food): void`**
- Public method: **`eatAll(Food[]): void`**

`Predator:`

- Private field: **`health: int`**
- Public method: **`feed(Food): void`**

`Food:`

- Just an empty class

Note: First, make **`eatAll()`** to use **`eat()`** to do its job. Do not make the **`eat()`** method **`final`**. **`override eat()`** in **`Predator`**. Now if you change the implementation of **`eatAll()`** (to no longer use **`eat()`**, you can use **`Collections.addAll()`**) you should observe a bug introduced in you program.

5. Random Array List

Create a **`RandomArrayList`** class that has all the functionality of an **`ArrayList`**.

Add additional function that **returns** and **removes** a random element from the list.

- Public method: **`getRandomElement(): Object`**

Hints

6. Stack of Strings

Create a class **`Stack`** which can store only strings and has the following functionality:

- Private field: **`data: ArrayList<String>`**
- Public method: **`push(String item): void`**
- Public method: **`pop(): String`**
- Public method: **`peek(): String`**
- Public method: **`isEmpty(): boolean`**

```
public static void main(String[] args) {
    StackOfStrings sos = new StackOfStrings();
    sos.push("one");
    sos.push("tow");
    sos.push("three");

    System.out.println(sos.isEmpty());
    System.out.println(sos.peek());

    System.out.println(sos.pop());
    System.out.println(sos.pop());
    System.out.println(sos.pop());
}
```

Hints

Use composition/delegation in order to have a field in which to store the stack's data