

# Lab: Inheritance

This document defines the lab for "[Java OOP](#)" course @ [Software University](#). Please submit your solutions (source code) of all below described problems in [Judge](#).

## Part I: Inheritance

### 1. Single Inheritance

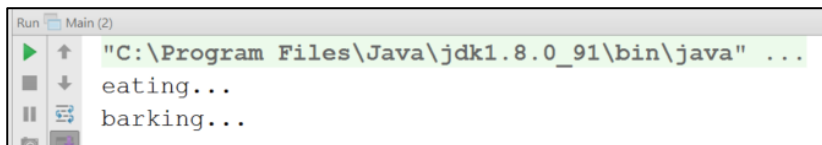
Create two classes named **Animal** and **Dog**.

**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**.

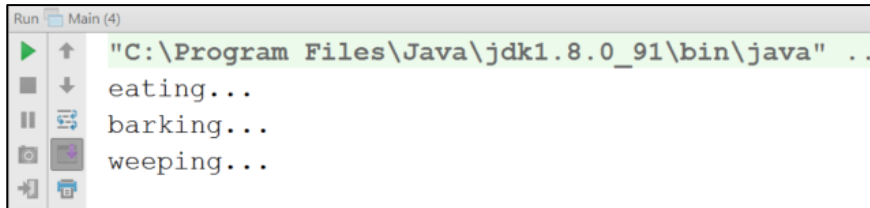
**Animal** with a single public method **eat()** that prints: "eating..."

**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..."

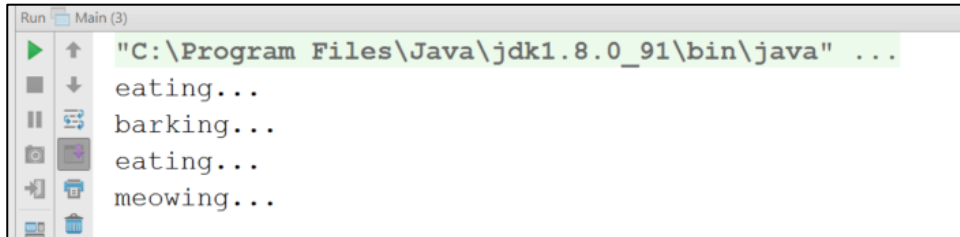
**Dog** and **Cat** should inherit from **Animal**.

```
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 II: Reusing Classes

### 4. 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**

### 5. 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("two");  
    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