iNTERACTIVE PLAYGROUND NOTES ON JAVA LAB

By Jason Pandian
Assistant Professor, Department of Information Technology

EXP2. Demonstrate various types of Inheritance in Java

1.Single Inheritance

AIM:

To demonstrate single inheritance where one class inherits from another.

ALGORITHM

- Step-1. Create a base class Animal with a method makeSound().
- Step-2. Create a derived class Dog that extends Animal and overrides the makeSound() method.
- Step-3. Create a Main class with a main() method to instantiate Dog and call makeSound().

PROGRAM

Animal.java

In [52]:
```
%%writefile src/singleinheritance/Animal.java
package singleinheritance;

public class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
```

```
Overwriting src/singleinheritance/Animal.java
```

Dog.java

In [53]:
```
%%writefile src/singleinheritance/Dog.java
package singleinheritance;

public class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Dog barks");
    }
}
```

```
Overwriting src/singleinheritance/Dog.java
```

SingleInheritanceMain

In [54]:
```
%%writefile src/singleinheritance/SingleInheritanceMain.java
package singleinheritance;

public class SingleInheritanceMain {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.makeSound(); // Output: Dog barks
    }
}
```

```
Overwriting src/singleinheritance/SingleInheritanceMain.java
```

Expected Output

Dog barks

Logical Organization

In [55]:
```
!tree
```

```
.
├── l1-introduction-to-java-lab.ipynb
├── l2-inheritance-to-java-lab copy.ipynb
└── src
    ├── CheckPrime.class
    ├── CheckPrime.java
    ├── SecondLargestNumber.class
    ├── SecondLargestNumber.java
    ├── multilevelinheritance
    │   ├── Animal.class
    │   ├── Animal.java
    │   ├── Dog.class
    │   ├── Dog.java
    │   ├── Mammal.class
    │   ├── Mammal.java
    │   ├── MultilevelInheritanceMain.class
    │   └── MultilevelInheritanceMain.java
    └── singleinheritance
        ├── Animal.class
        ├── Animal.java
        ├── Dog.class
        ├── Dog.java
        ├── SingleInheritanceMain.class
        └── SingleInheritanceMain.java

4 directories, 20 files
```

Regular Java Complier(Linux)

```
nit@linux$: cd src
nit@linux:~..../src$: javac singleinheritance/*.java
nit@linux:~..../src$: java singleinheritance.SingleInheritanceMain
```

JUPYTER NOTEBOOK

In [56]:
```
!javac  ../lectures/src/singleinheritance/*.java
!java -cp ../lectures/src singleinheritance.SingleInheritanceMain
```

```
Dog barks
```

2. Multilevel Inheritance

AIM:

To demonstrate multilevel inheritance where a class inherits from another class, which itself inherits from another class.

ALGORITHM

- Step-1. Create a base class Animal.

- Step-2. Create a subclass Mammal that extends Animal.
- Step-3. Create a subclass Dog that extends Mammal.
- Step-4. In the Main class, instantiate Dog and call its methods.

PROGRAM

Animal.java

In [57]:
```
%%writefile src/multilevelinheritance/Animal.java

package multilevelinheritance;

public class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
```

```
Overwriting src/multilevelinheritance/Animal.java
```

Mammal.java

In [58]:
```
%%writefile src/multilevelinheritance/Mammal.java

package multilevelinheritance;

public class Mammal extends Animal {
    public void hasFur() {
        System.out.println("Mammal has fur");
    }
}
```

```
Overwriting src/multilevelinheritance/Mammal.java
```

Dog.java

In [59]:
```
%%writefile src/multilevelinheritance/Dog.java


package multilevelinheritance;

public class Dog extends Mammal {
    @Override
    public void makeSound() {
        System.out.println("Dog barks");
    }
}
```

```
Overwriting src/multilevelinheritance/Dog.java
```

In [60]:
```
%%writefile src/multilevelinheritance/MultilevelInheritanceMain.java

package multilevelinheritance;
```

```java
public class MultilevelInheritanceMain {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.makeSound(); // Output: Dog barks
        dog.hasFur();    // Output: Mammal has fur
    }
}
```

```
Overwriting src/multilevelinheritance/MultilevelInheritanceMain.java
```

Expected Output

Dog barks

Mammal has fur

Logical Organization

In [50]:
```
!tree
```

```
.
├── l1-introduction-to-java-lab.ipynb
├── l2-inheritance-to-java-lab copy.ipynb
└── src
    ├── CheckPrime.class
    ├── CheckPrime.java
    ├── SecondLargestNumber.class
    ├── SecondLargestNumber.java
    ├── multilevelinheritance
    │   ├── Animal.class
    │   ├── Animal.java
    │   ├── Dog.class
    │   ├── Dog.java
    │   ├── Mammal.class
    │   ├── Mammal.java
    │   ├── MultilevelInheritanceMain.class
    │   └── MultilevelInheritanceMain.java
    └── singleinheritance
        ├── Animal.class
        ├── Animal.java
        ├── Dog.class
        ├── Dog.java
        ├── SingleInheritanceMain.class
        └── SingleInheritanceMain.java

4 directories, 20 files
```

Regular Java Complier(Linux)

```
nit@linux:~..../src$: javac multilevelinheritance/*.java
nit@linux:~..../src$: java
multilevelinheritance.MultilevelInheritanceMain
```

In [51]:
```
!javac  ../lectures/src/multilevelinheritance/*.java
!java -cp ../lectures/src multilevelinheritance.MultilevelInheritanceMain
```

```
Dog barks
Mammal has fur
```

3. Hierarchical Inheritance

AIM:

To demonstrate hierarchical inheritance where multiple classes inherit from a single base class.

ALGORITHM

- Step-1. Create a base class Animal.
- Step-2. Create multiple subclasses Dog and Cat that extend Animal.
- Step-3. In the Main class, instantiate Dog and Cat, and call their methods.

PROGRAM

Animal.java

In [66]:
```
%%writefile src/hierarchicalinheritance/Animal.java


package hierarchicalinheritance;

public class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
```

```
Overwriting src/hierarchicalinheritance/Animal.java
```

Dog.java

In [67]:
```
%%writefile src/hierarchicalinheritance/Dog.java


package hierarchicalinheritance;

public class Dog extends Animal {
    @Override
```

```
    public void makeSound() {
        System.out.println("Dog barks");
    }
}
```

```
Overwriting src/hierarchicalinheritance/Dog.java
```

Cat.java

In [68]:
```
%%writefile src/hierarchicalinheritance/Cat.java

package hierarchicalinheritance;

public class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Cat meows");
    }
}
```

```
Overwriting src/hierarchicalinheritance/Cat.java
```

HierarchicalInheritanceMain.java

In [69]:
```
%%writefile src/hierarchicalinheritance/HierarchicalInheritanceMain.java


package hierarchicalinheritance;

public class HierarchicalInheritanceMain {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.makeSound(); // Output: Dog barks

        Cat cat = new Cat();
        cat.makeSound(); // Output: Cat meows
    }
}
```

```
Writing src/hierarchicalinheritance/HierarchicalInheritanceMain.java
```

Expected Output

Dog barks

Cat meows

Logical Organization

In [73]:
```
!tree
```

```
.
├── l1-introduction-to-java-lab.ipynb
├── l2-inheritance-to-java-lab copy.ipynb
└── src
    ├── CheckPrime.class
    ├── CheckPrime.java
    ├── SecondLargestNumber.class
    ├── SecondLargestNumber.java
    ├── hierarchicalinheritance
    │   ├── Animal.class
    │   ├── Animal.java
    │   ├── Cat.class
    │   ├── Cat.java
    │   ├── Dog.class
    │   ├── Dog.java
    │   ├── HierarchicalInheritanceMain.class
    │   └── HierarchicalInheritanceMain.java
    ├── multilevelinheritance
    │   ├── Animal.class
    │   ├── Animal.java
    │   ├── Dog.class
    │   ├── Dog.java
    │   ├── Mammal.class
    │   ├── Mammal.java
    │   ├── MultilevelInheritanceMain.class
    │   └── MultilevelInheritanceMain.java
    └── singleinheritance
        ├── Animal.class
        ├── Animal.java
        ├── Dog.class
        ├── Dog.java
        ├── SingleInheritanceMain.class
        └── SingleInheritanceMain.java

5 directories, 28 files
```

Regular Java Complier(Linux)

```
nit@linux:~..../src$: javac hierarchicalinheritance/*.java
nit@linux:~..../src$: java
hierarchicalinheritance.HierarchicalInheritanceMain
```

In [74]:
```
!javac  ../lectures/src/hierarchicalinheritance/*.java
!java -cp ../lectures/src hierarchicalinheritance.HierarchicalInheritanceMai
```

```
Dog barks
Cat meows
```

4. Multiple Inheritance (via Interfaces)

AIM:

To demonstrate multiple inheritance using interfaces, where a class implements multiple interfaces.

ALGORITHM

- Step-1. Define two interfaces Flyable and Swimmable with methods fly() and swim().
- Step-2. Create a class Duck that implements both interfaces.
- Step-3. In the Main class, instantiate Duck and call its methods.

PROGRAM

Flyable.java

In [82]:
```
%%writefile src/multipleinheritance/Flyable.java

package multipleinheritance;

public interface Flyable {
    void fly();
}
```

```
Overwriting src/multipleinheritance/Flyable.java
```

Swimmable.java

In [76]:
```
%%writefile src/multipleinheritance/Swimmable.java

package multipleinheritance;

public interface Swimmable {
    void swim();
}
```

```
Writing src/multipleinheritance/Swimmable.java
```

Duck.java

In [79]:
```
%%writefile src/multipleinheritance/Duck.java

package multipleinheritance;

public class Duck implements Flyable, Swimmable {
    @Override
    public void fly() {
        System.out.println("Duck flies");
    }

    @Override
    public void swim() {
        System.out.println("Duck swims");
```

```
    }
}
```

```
Overwriting src/multipleinheritance/Duck.java
```

MultipleInheritanceMain.java

In [78]:
```
%%writefile src/multipleinheritance/MultipleInheritanceMain.java


package multipleinheritance;

public class MultipleInheritanceMain {
    public static void main(String[] args) {
        Duck duck = new Duck();
        duck.fly();  // Output: Duck flies
        duck.swim(); // Output: Duck swims
    }
}
```

```
Writing src/multipleinheritance/MultipleInheritanceMain.java
```

Expected Output

Duck flies

Duck swims

Logical Organization

In [80]:
```
!tree
```

```
.
├── l1-introduction-to-java-lab.ipynb
├── l2-inheritance-to-java-lab copy.ipynb
└── src
    ├── CheckPrime.class
    ├── CheckPrime.java
    ├── SecondLargestNumber.class
    ├── SecondLargestNumber.java
    ├── hierarchicalinheritance
    │   ├── Animal.class
    │   ├── Animal.java
    │   ├── Cat.class
    │   ├── Cat.java
    │   ├── Dog.class
    │   ├── Dog.java
    │   ├── HierarchicalInheritanceMain.class
    │   └── HierarchicalInheritanceMain.java
    ├── multilevelinheritance
    │   ├── Animal.class
    │   ├── Animal.java
    │   ├── Dog.class
    │   ├── Dog.java
    │   ├── Mammal.class
    │   ├── Mammal.java
    │   ├── MultilevelInheritanceMain.class
    │   └── MultilevelInheritanceMain.java
    ├── multipleinheritance
    │   ├── Duck.java
    │   ├── Flyable.java
    │   ├── MultipleInheritanceMain.java
    │   └── Swimmable.java
    └── singleinheritance
        ├── Animal.class
        ├── Animal.java
        ├── Dog.class
        ├── Dog.java
        ├── SingleInheritanceMain.class
        └── SingleInheritanceMain.java

6 directories, 32 files
```

Regular Java Complier(Linux)

```
nit@linux:~..../src$: javac multipleinheritance/*.java
nit@linux:~..../src$: java
multipleinheritance.MultipleInheritanceMain
```

In [81]:
```
!javac  ../lectures/src/multipleinheritance/*.java
!java -cp ../lectures/src multipleinheritance.MultipleInheritanceMain
```

```
Duck flies
Duck swims
```

🎉 *Congratulations on Completing Your Java ☕ Inheritance Lab!*