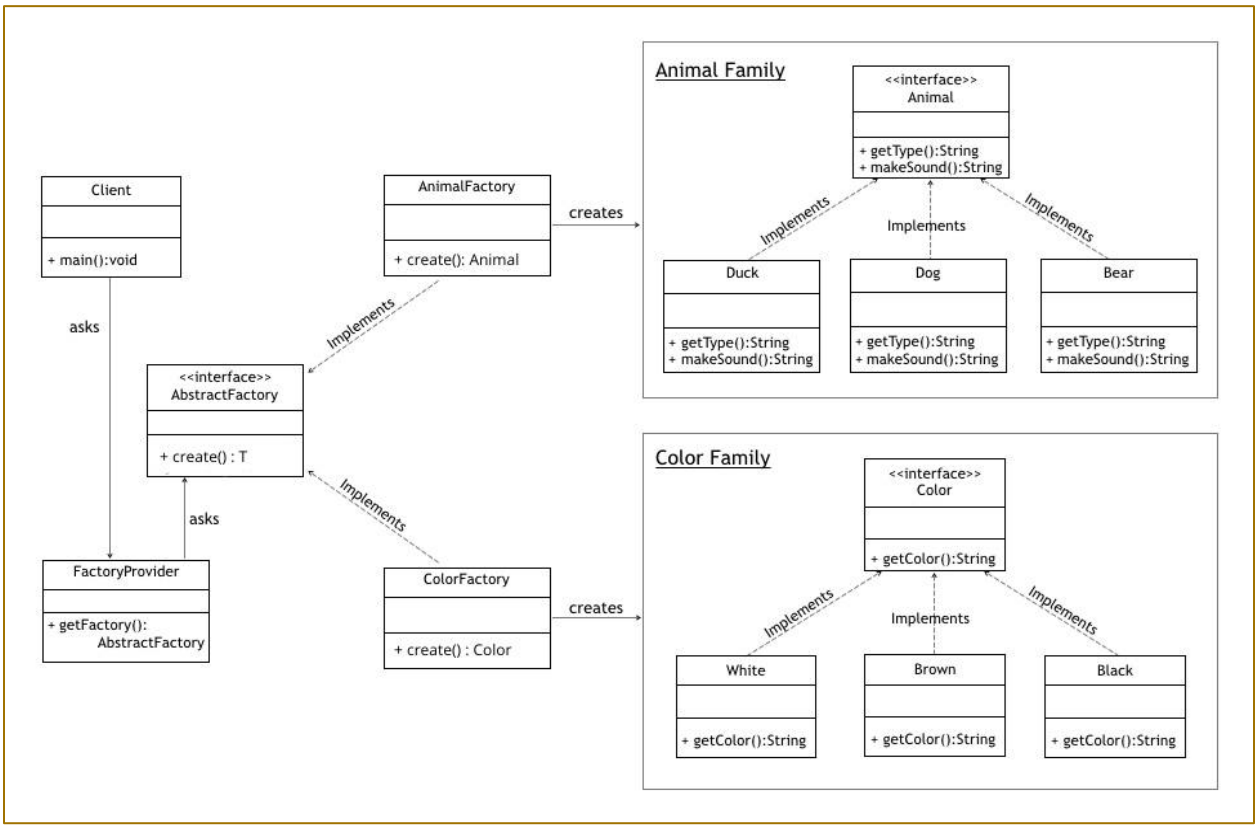
CPIT252

# Lab Activity #04

Elaf Yousef Aloufi, 1911265 - CAR

## Class Diagram



# Code

Client Class (main)

```
public class Client {

    public static void main(String[] args) {

        System.out.println("\n-----------------------This is the main class-----------------------");

        FactoryProvider factoryProvider = new FactoryProvider();

        AbstractFactory animal = factoryProvider.getFactory(true);

        Animal dog = animal.create("Dog");
        System.out.println(dog.getType());
        System.out.println(dog.makeSound()+"\n");

        Animal duck = animal.create("Duck");
        System.out.println(duck.getType());
        System.out.println(duck.makeSound()+"\n");

        Animal bear = animal.create("Bear");
        System.out.println(bear.getType());
        System.out.println(bear.makeSound()+"\n");

        System.out.println("-------------------------------------------------------------------------");

        AbstractFactory color = factoryProvider.getFactory(false);

        Color black = color.create("Black");
        System.out.println(black.getCololr()+"\n");

        Color brown = color.create("Brown");
        System.out.println(brown.getCololr()+"\n");

        Color white = color.create("White");
        System.out.println(white.getCololr()+"\n");
```

## FactoryProvider Class

```java
public class FactoryProvider {

    public AbstractFactory getFactory(boolean rounded) {
        System.out.println("\nInside getFactory() method in the FactoryProvider class\n");
        if (rounded) {
            return new AnimalFactory();
        } else {
            return new ColorFactory();
        }
    }


}
```

## AbstractFactory Class

```java
public interface AbstractFactory {

    public abstract <T> T create(String type);

}
```

## AnimalFactory Class

```java
public class AnimalFactory implements AbstractFactory {

    @Override
    public Animal create(String animalType) {
        if (animalType.equalsIgnoreCase("Dog")) {
            System.out.println("Inside create() method in AnimalFactory class [Dog object is created]");
            return new Dog();
        } else if (animalType.equalsIgnoreCase("Duck")) {
            System.out.println("Inside create() method in AnimalFactory class [Duck object is created]");
            return new Duck();
        } else if (animalType.equalsIgnoreCase("Bear")) {
            System.out.println("Inside create() method in AnimalFactory class [Bear object is created]");
            return new Bear();
        }
        return null;

    }


}
```

## Animal Class

```
public interface Animal {
    public String getType();
    public String makeSound();

}
```

## Dog Class

```
public class Dog implements Animal {

    @Override
    public String getType() {
        return "Type: Dog";

    }

    @Override
    public String makeSound() {
        return "Sound: Howl";

    }
}
```

## Duck Class

```
public class Duck implements Animal {

    @Override
    public String getType() {
        return "Type: Duck";

    }

    @Override
    public String makeSound() {
        return "Sound: Quack";

    }
}
```

## Bear Class

```
public class Bear implements Animal{
        @Override
        public String getType(){
         return "Type: Bear";

      }
        @Override
    public String makeSound(){
         return "Sound: Roar";

    }
}
```

## ColorFactory Class

```
public class ColorFactory implements AbstractFactory {

    @Override
    public Color create(String colorType) {
        if (colorType.equalsIgnoreCase("Black")) {
            System.out.println("Inside create() method in ColorFactory class [Black object is created]");
            return new Black();
        } else if (colorType.equalsIgnoreCase("Brown")) {
            System.out.println("Inside create() method in ColorFactory class [Brown object is created]");
            return new Brown();
        } else if (colorType.equalsIgnoreCase("White")) {
            System.out.println("Inside create() method in ColorFactory class [White object is created]");
            return new White();
        }
        return null;

    }

}
```

## Color Class

```
public interface Color {
    public String getCololr();
}
```

Black Class

```
public class Black implements Color{
    @Override
    public String getCololr(){
        return "Color: Black";

    }
}
```

Brown Class

```
public class Brown implements Color{
    @Override
    public String getCololr(){
        return "Color: Brown";

    }
}
```

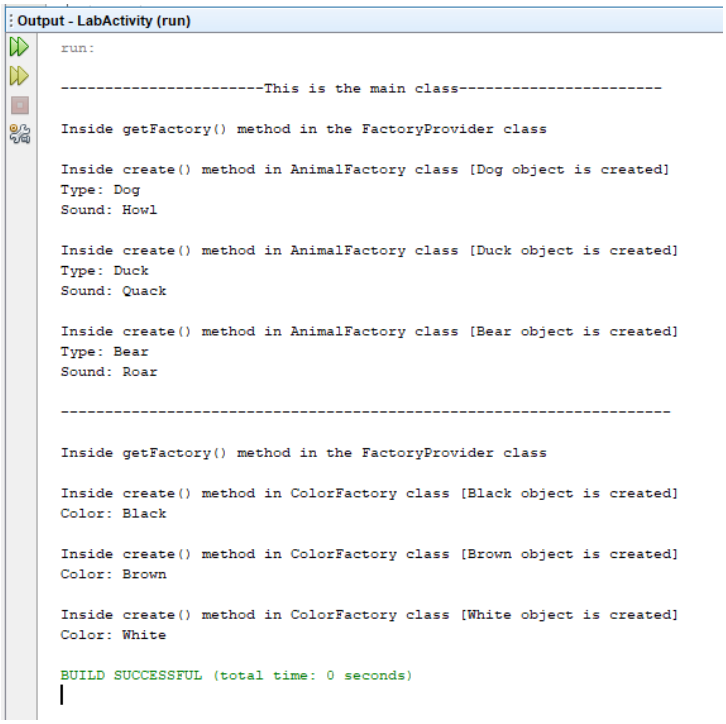
White Class

```
public class White implements Color {

    @Override
    public String getCololr() {
        return "Color: White";

    }
}
```

# Output


Output - LabActivity (run)
run:
----------This is the main class----------
Inside getFactory() method in the FactoryProvider class
Inside create() method in AnimalFactory class [Dog object is created]
Type: Dog
Sound: Howl
Inside create() method in AnimalFactory class [Duck object is created]
Type: Duck
Sound: Quack
Inside create() method in AnimalFactory class [Bear object is created]
Type: Bear
Sound: Roar
Inside getFactory() method in the FactoryProvider class
Inside create() method in ColorFactory class [Black object is created]
Color: Black
Inside create() method in ColorFactory class [Brown object is created]
Color: Brown
Inside create() method in ColorFactory class [White object is created]
Color: White
BUILD SUCCESSFUL (total time: 0 seconds)