DESIGN PATTERNS

Design pattern ???

1. Repeatable problem
2. Common solution
3. Can’t be transformed directly into code

It is a description or template for how to solve a problem that can be used in many different situations.

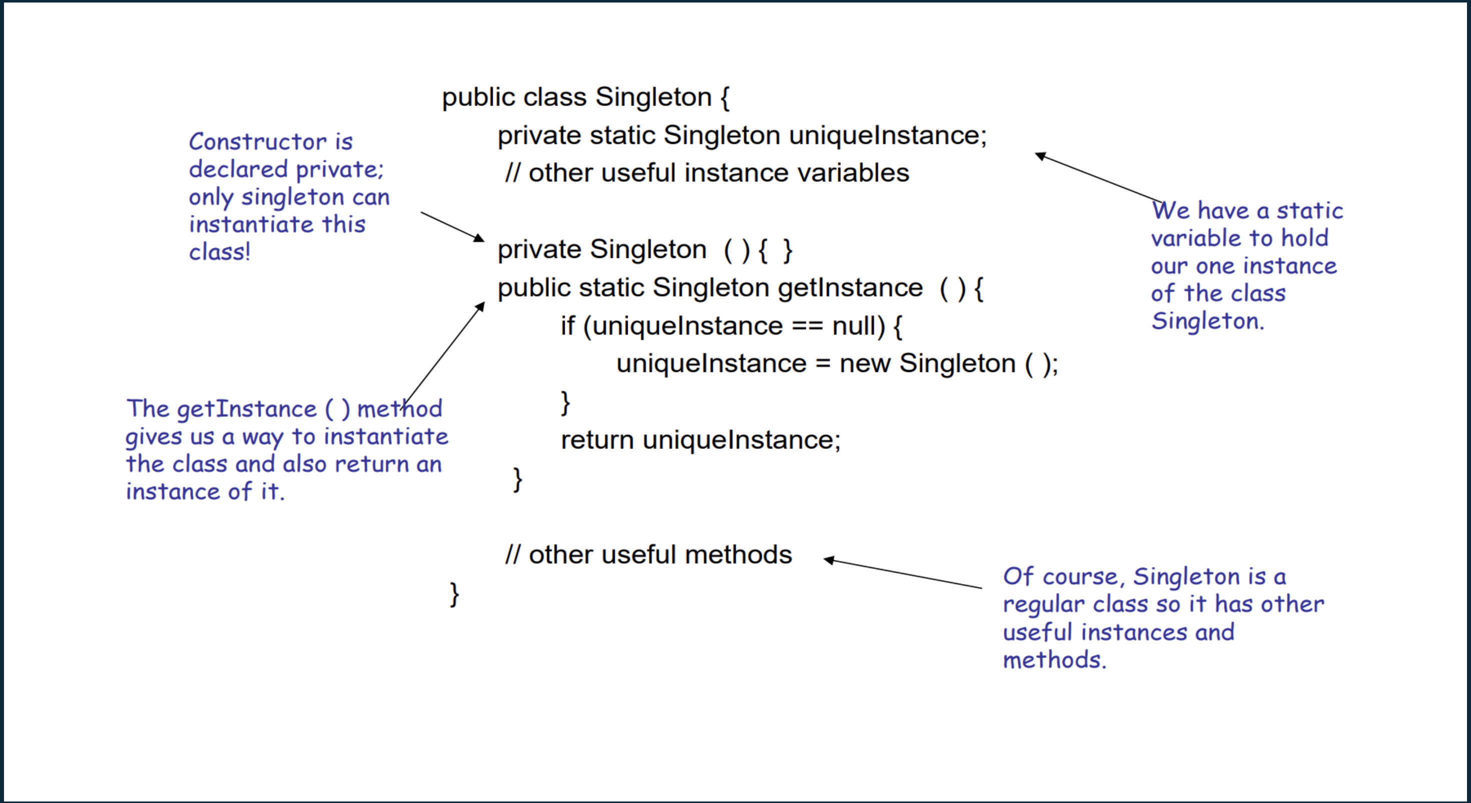
The Singleton Pattern

The Singleton Pattern ensures a class has only one instance, and provides a global point of access to it.

Why ???

- Many objects we need only one of: dialog boxes, objects that handle preferences
- If more than one instantiated => Incorrect program behavior, overuse of resources, inconsistent results.

Implementation



```
public class Singleton {
      private static Singleton uniqueInstance;
       // other useful instance variables

      private Singleton  ( ) {  }
      public static Singleton getInstance  ( ) {
             if (uniqueInstance == null) {
                    uniqueInstance = new Singleton ( );
             }
             return uniqueInstance;
       }

       // other useful methods
}
```

Constructor is declared private; only singleton can instantiate this class!

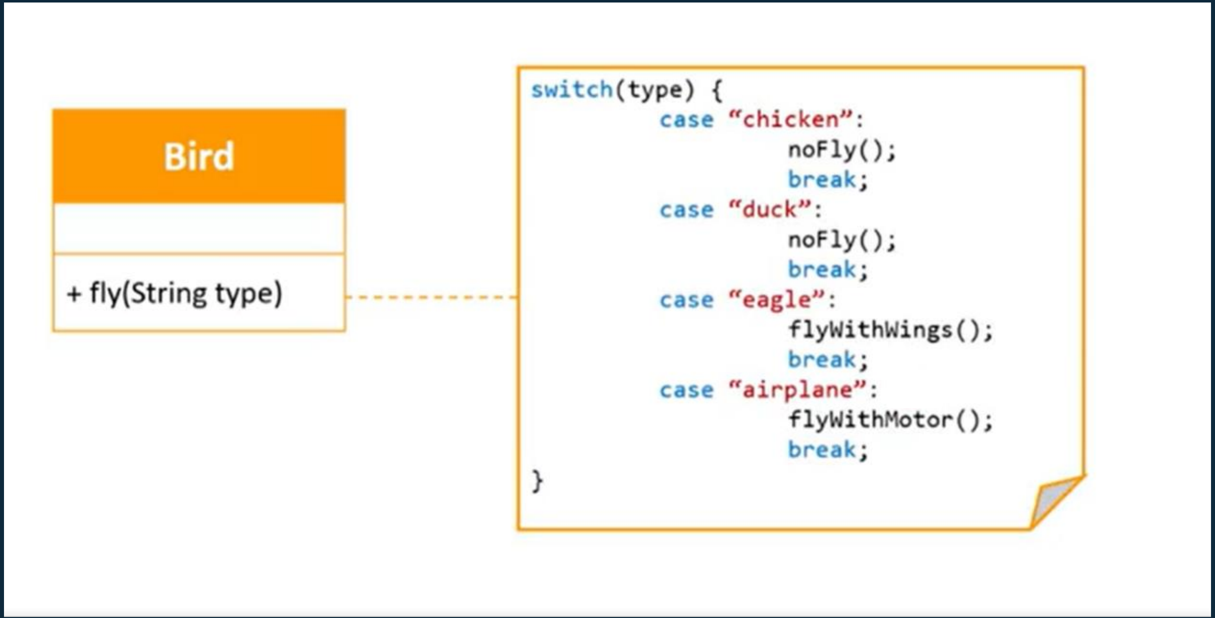
The getInstance ( ) method gives us a way to instantiate the class and also return an instance of it.

We have a static variable to hold our one instance of the class Singleton.

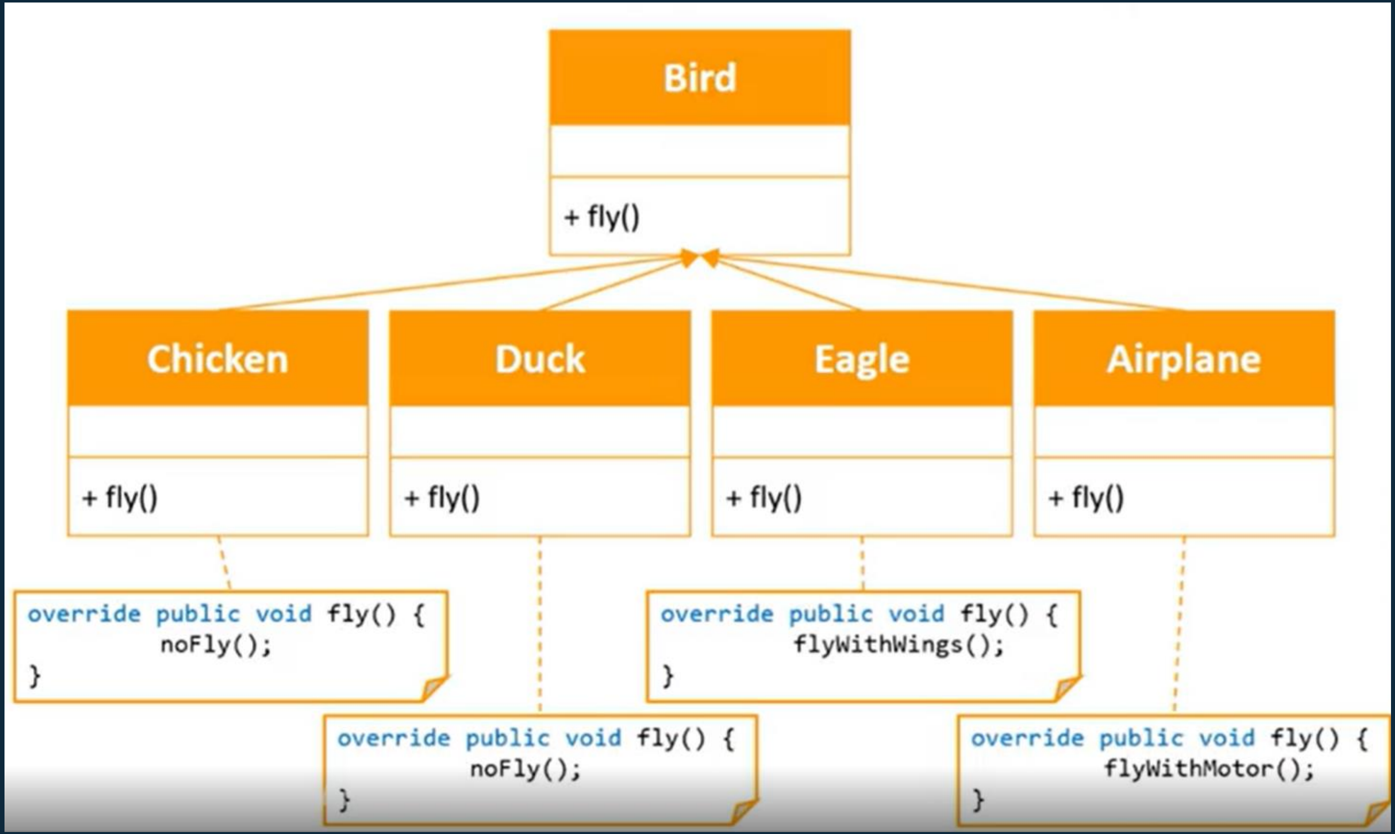
Of course, Singleton is a regular class so it has other useful instances and methods.

Strategy Pattern

Strategy is a behavioral design pattern that lets you define a family of algorithms, put each of them into a separate class, and make their objects interchangeable.



```
switch(type) {
        case "chicken":
                noFly();
                break;
        case "duck":
                noFly();
                break;
        case "eagle":
                flyWithWings();
                break;
        case "airplane":
                flyWithMotor();
                break;
}
```

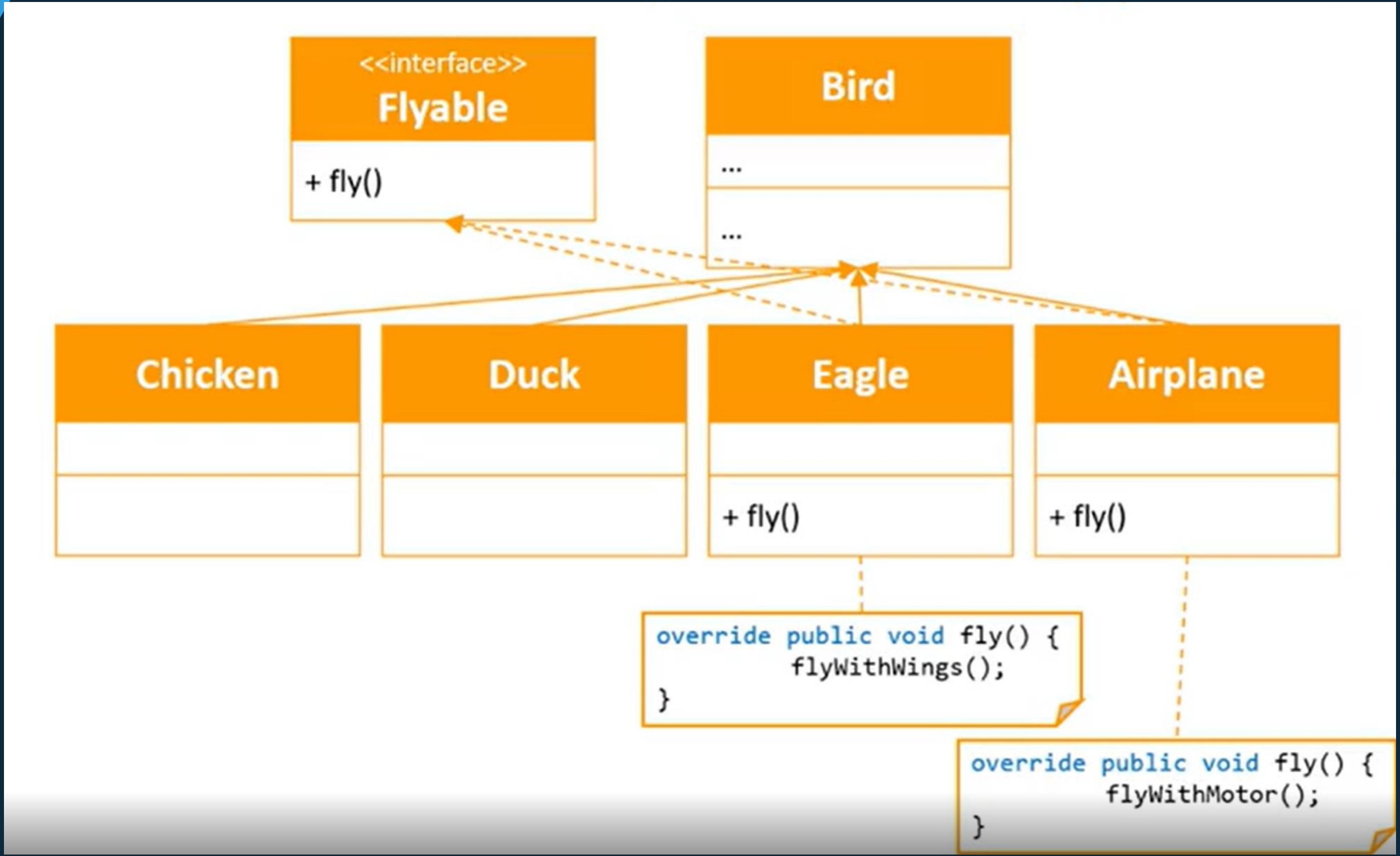


```
override public void fly() {
        noFly();
}
```

```
override public void fly() {
        noFly();
}
```

```
override public void fly() {
        flyWithWings();
}
```

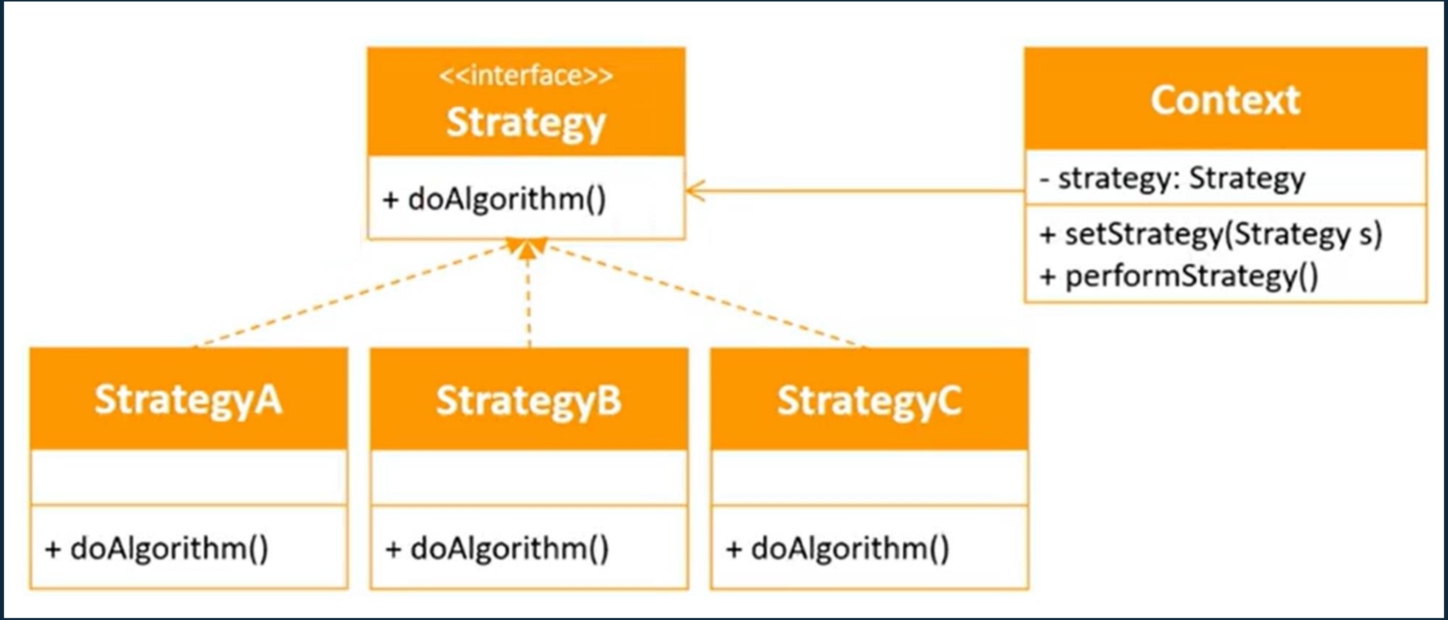
```
override public void fly() {
        flyWithMotor();
}
```



```
override public void fly() {
        flyWithWings();
}
```

```
override public void fly() {
        flyWithMotor();
}
```

Final Solution



عید تون مبارک باشه