**1. Static Class**

* We can declare a class as static if and only if it is a nested class.
* In the case of normal or regular class without the existing outer class object, there is no chance of an existing inner class object i.e inner class object is strongly associated with an outer class object.
* In the case of static nested classes, there is a way of declaring the nested class object without an outer class object.

**Properties of the static class:**

1. For a static class objects cannot be created.
2. A static class can only have static members.
3. A static class cannot access members (non-static) of the outer class.

### 2. Final Class

* A class can be declared as final by using the keyword ‘final’.
* If a class is declared as final then it can’t be extended by any other class, can’t create a child class for that class, and inheritance is not possible for final classes.
* Every method present inside the final class is usually final by default, but every variable present inside the ultimate class need not be final.
* On creating a final class then it cannot achieve inheritance.
* On creating a final method then it cannot achieve polymorphism, but we can obtain security.
* No one can change the code’s unique implementation,it is primarily used to prevent inheritance and polymorphism.

### 3. Abstract Class

* A class can be declared as abstract by using the keyword ‘abstract’.
* For any java class if there's at least one abstract method then objects are not allowed to be created of such type of class,
* Should declare the class with the abstract modifier, i.e
* For abstract classes instantiation is not possible.
* A class containing either zero or more abstract methods can be an abstract class if we don’t want any instantiation
* If an abstract class is extended then for each and every abstract method of parent class implementation should be provided, otherwise, child class should be declared as abstract. In this case, the next-level child class is responsible for providing the implementation.

### 4. Concrete Class

A concrete class is a normal or regular class in java. A concrete class is a class that extends another class or implements an interface. In short, we can say that any class which is not abstract is said to be a concrete class. We can directly create an object for the concrete class.

***Note****: A class is said to be a concrete class if there is an implementation for each and every method.*

**5. Singleton Class**

A singleton class is a class that can have only one object (an instance of the class) at a time.

**Advantages**:

* The primary purpose of a java Singleton class is to restrict the limit of the number of object creations to only one. This often ensures that there is access control to resources, for example, socket or database connection.
* Memory space wastage does not occur with the use of the singleton class because it restricts instance creation. As the object creation will take place only once instead of creating it each time a new request is made.

We can also create our own singleton classes, for that we’d like to possess

* Private constructor
* Private static variable and the public factory method

**6. POJO Class**

“Plain Old Java Object” is an abbreviation for “Plain Old Java Object.” A POJO class has only private variables with setter and getter methods to access them. It’s a pure data structure with fields that can override some Object methods (e.g. equals) or other interfaces (e.g. serializable), but it has no behavior of its own.It provides Encapsulation.

**Properties of POJO class:**

* Class must be declared as public
* When writing a POJO class, public setter and getter methods are required.
* Private variables should be used for all instance variables.
* It should not extend classes that have already been defined.
* It should not implement interfaces that have been pre-defined.

### 7. Inner Class

Sometimes we can declare a class inside another class. Such types of classes are called inner classes. Inner classes concept is introduced to fix GUI bugs as a part of event handling but because of powerful features and benefits of inner classes slowly programmers have started using it in regular coding as well. Without existing one type of object there is no chance of executing another type of object then we should go for inner classes.

**Example**: University consists of several departments, without an existing university there is no chance of an existing department, hence we have to declare department class inside university class.

Class University{

// code

Class Department{

// code

}

}

The relation between outer class and inner class is said to be a Has-A relationship. Based on the position of declaration and behavior all inner classes are divided into four types.

* Normal or Regular classes
* Method local Inner classes
* Anonymous Inner classes(are used against functional interface)
* Static nested classes