**1. Static Class**

We can declare a class as static if and only if it is a nested class. We can declare an inner class with the static modifier, such types of inner classes are called static nested classes. 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 may be a chance of declaring the nested class object without an outer class object.

**Properties of the static class:**

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

### 2. Final Class

The class can be declared as final by using the keyword ‘final’. If a class is declared as final we can’t extend the functionality of that class i.e we can’t create a child class for that class, i.e inheritance is not possible for final classes. Every method present inside the ultimate class is usually final by default, but every variable present inside the ultimate class needn’t be final. If we create the final class we cannot achieve inheritance. If we create a final method we cannot achieve polymorphism, but we can obtain security. No one can change our code’s unique implementation, but the main disadvantage of the final keyword is we are missing the benefits of inheritance and polymorphism.

### 3. Abstract Class

We can declare a class as abstract by using the keyword ‘abstract’. For any java class if we are not allowed to create an object of such type of class, we should declare it with the abstract modifier, i.e for abstract classes instantiation is not possible. If a class contains at least one abstract method then we should declare a class as abstract otherwise we get a compilation error. **Reason**: If a class contains at least one abstract method then implementation is not complete and hence it is not recommended to create an object. If we need to restrict object instantiation compulsory we need to declare the class as abstract. A class containing either zero or more abstract methods can be an abstract class if we don’t want any instantiation

If we are extending abstract class then for each and every abstract method of parent class we should always provide implementation, otherwise, we’ve to declare child class 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**

For any java class if we are allowed to make just one object such sort of class is claimed to be a singleton class.

**Example**: Runtime, BusinessDelegate, ServiceLocator

**Advantages**:

* If several people have the same requirement then it’s not recommended to make a separate object for each requirement, we’ve to make just one object so that we can reuse an equivalent object for each similar requirement, in order that performance and memory utilization are going to be improved.
* This is often the central idea of singleton classes.

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**

POJO stands for Plain Old Java Object. If we write a category, then it got to follow some rules referred to as POJO rules. it’s wont to java to extend readability and reusability. It provides Encapsulation.

**Properties of POJO class:**

1. class must be declared as public
2. properties/variables must be declared as private
3. Must have a public default constructor
4. May or might not have argument constructor
5. Every property should have a public getter and setter methods
6. It cannot contain pre-specified annotations.

### 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 are started using in regular coding also. 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