|  |  |
| --- | --- |
| **Constant** | **Variable** |
| A constant does not change its value over time.  Constants are usually written in numbers.  Constants usually represent the known values in an equation, expression or in line of programming. | variable, on the other hand, changes its value dependent on the equation  Variables are specially written in letters or symbols.  Variables, on the other hand, represent  the unknown values. |

**Static Method**

* particular piece of code is to be shared by all the instance methods.

**Limitation of Static Method And Static Block**

* **y**ou cannot access a non-static member (method or, variable) from a static context.
* This and super cannot be used in static context.
* The static method can access only static type data (static type instance variable).
* You cannot override a static method.

**Static blocks**

* You cannot invoke a static block explicitly.
* If exception occurs in a static block you must wrap it within try-catch pair. You cannot throw it.
* You cannot use this and *super*keywords inside a static block.
* You cannot control the order of execution dynamically in case of static blocks, they will be executed in the order of their declaration.

**Abstract Class and when to use**

* An abstract class is also good if we want to declare non-public members.
* If we want to add new methods in the future, then an abstract class is a better choice.
* If we want to create multiple versions of our component, create an abstract class. Abstract classes provide a simple and easy way to version our components. By updating the base class, all inheriting classes are automatically updated with the change.
* Abstract classes allow us to partially implement our class

**Abstract Class can be Final**

* An abstract class is incomplete and can only be instantiated by extending a concrete class and implementing all abstract methods, while a final class is considered complete and cannot be extended further.
* This means when you make an abstract class final, it cannot be extended hence it cannot be used and that's why the Java compiler throws a compile-time error when you try to make an abstract class final in Java

**Can Class Extend More Than One class**

No because it creates diamond problem, jvm does not know which superclass constructor to call and which method should be executed