**Abstraction Vs Interfaces**

* Interface never does implementation of any method. method is always implemented in the child class.
* Static method not allowed. Because we override those methods in the child class.
* A **Single class** can inherit the properties from **multiple Interfaces**
* If we declare method as static Warning: Static methods cannot be annotated with @Override
* Variable are static by Default
* Warning when trying to create Object **'ICICIBank' is abstract; cannot be instantiated,** We cannot create the Object of Interface.
* If we call Interface variable without className :

**import static abstractionConcept.RBI\_Bank.*min\_balance*;**

* We can use Interface as reference variable for the child class Object

RBIBank rb = new KotakMahindra();

Interface have 100% abstraction

Abstraction

1. Abstract class must be defined with abstract Keyword.
2. Abstract class can have both types of method abstract and non-abstract
3. Method which defined with abstract keyword is never implemented.
4. Abstract class object never created.
5. We can define constructor in abstract class
6. Whenever we implementing constructor of child class, then first parent abstract class constructor gets executed.
7. We can achieve 0-100% abstraction in abstract class

|  |  |  |
| --- | --- | --- |
| No | Interface | Abstract Class |
| 1 | We only declare method in the interface. | Abstract class can have both abstract and non-abstract method which we can implement in the abstract class itself. |
| 2 | Interface does not have constructor | Abstract class have constructor |
| 3 | In Interface we can achieve 100% abstraction | Abstract class we can achieve 0 -100 % abstraction. |
| 4 | Variable are static by nature | Variable are non-static by default |
| 5 | Interface support multiple Inheritance | Abstract class does not support multiple Inheritance |
| 6 | The **interface keyword** is used to declare interface. | The **abstract keyword** is used to declare abstract class. |
| 7 | An **interface** can extend another Java interface only. | An **abstract class** can extend another Java class and implement multiple Java interfaces |
| 8 | An **interface** can be implemented using keyword "implements" | An **abstract class** can be extended using keyword "extends". |
| 9 | Members of a Java interface are public by default. | A Java **abstract class** can have class members like private, protected, etc. |

Common

1. We cannot create Object of Interface OR Abstract Class
2. We can create reference variable of Interface or abstract class which point to child class in which we implement methods