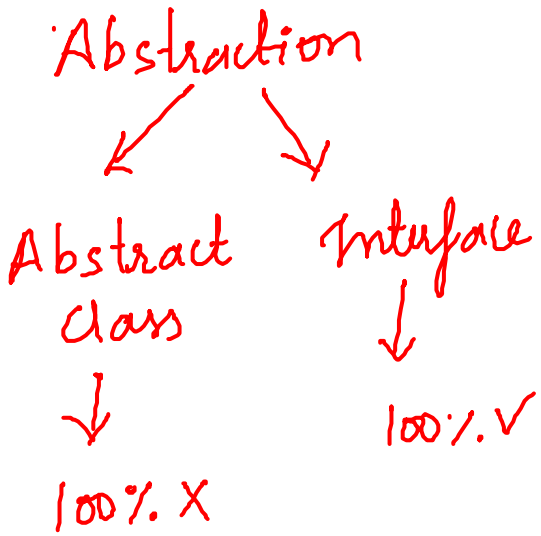
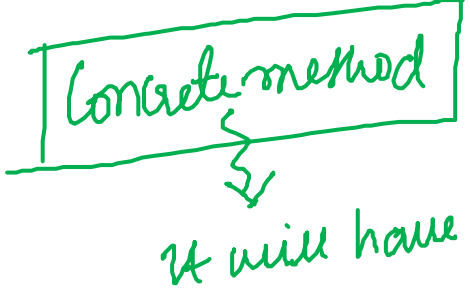
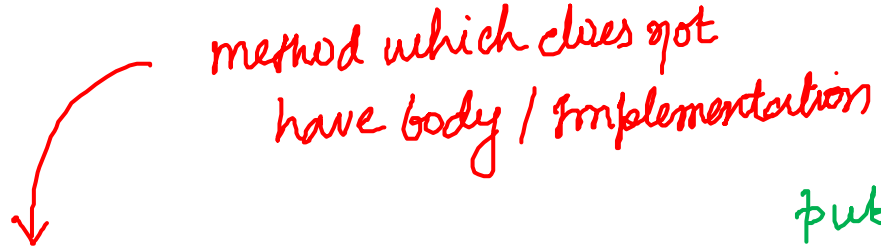
Interface

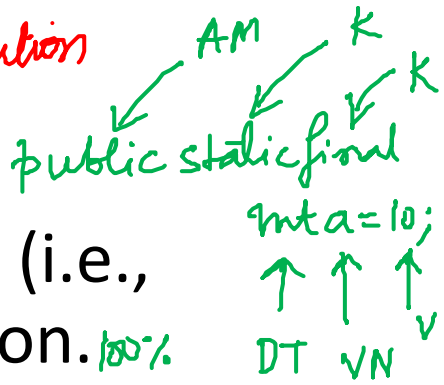
Praveen bhosle



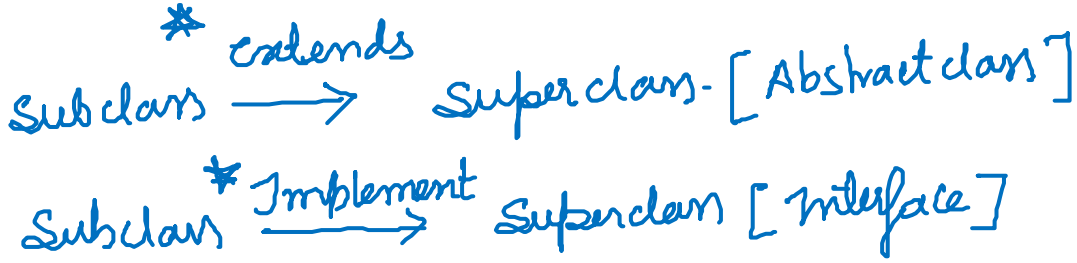
Interface







- An interface is a collection of abstract methods and constants (i.e., static and final fields). It is used to achieve complete abstraction.
- Note: Every interface in java is abstract by default. So, it is not compulsory to write abstract keyword with an interface.
- Note: A class that implements an interface is called implementation class. A class can implement any number of interfaces in Java.*



Syntax for interface

Allowed Access modifier for Interface
① default
② public
only

identifier
name

- <Access specifiers> interface <interface_name> {
- // declare constant fields ✓
- // declare methods that abstract ✓
- }

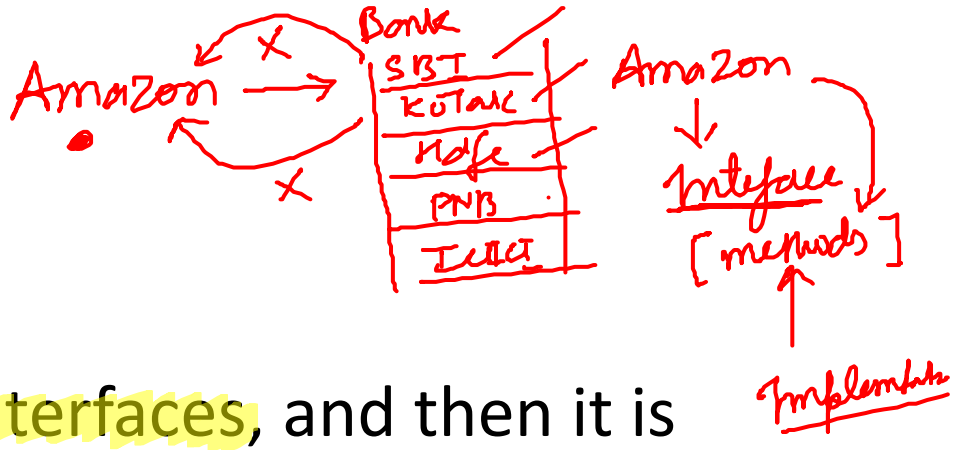
Features of interface

- 1. It contains public abstract methods and public static final variables by default.
- 2. We must follow I to C design principle in java. It means every class must be implemented by some interfaces.
- 3. In company, Team Lead or Manager level people can design the interface then give it to developer for implementing it.
- 4. Before 1.7, interface does not have any method body.
- 5. 1.8 Declare the default & static method with body in interface.
- 6. 1.9 we can define the private methods in interface also.
- 7. We cannot create the object of interface.
- 8. In interface, we can just define the method only but implemented those methods into implemented class.
- 9. Java supports multiple inheritance in the terms of interfaces but not classes.
- 10. Interface does not have constructor.

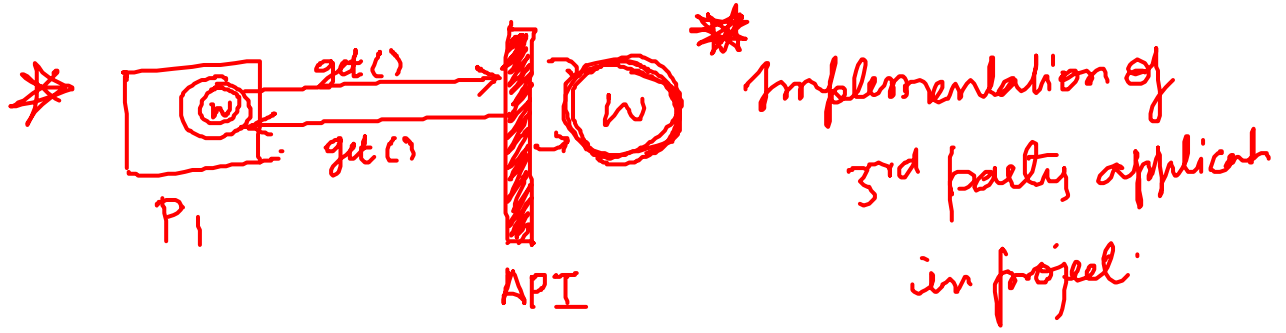
- Note:
- a) Earlier to Java 8, an interface could not define any implementation whatsoever. An interface can only declare abstract methods.
- b) Java 8 changed this rule. From Java 8 onwards, it is also possible to add a default implementation to an interface method.

default & static method

Why do we use Interface?



- 1. In industry, architect-level people create interfaces, and then it is given to developers for writing classes by implementing interfaces provided.
- Using interfaces is the best way to expose our project's API to some other projects. In other words, we can provide interface methods to the third-party vendors for their implementation.
- For example, HDFC bank can expose methods or interfaces to various shopping carts.



Coding…..

Relation between class and interface

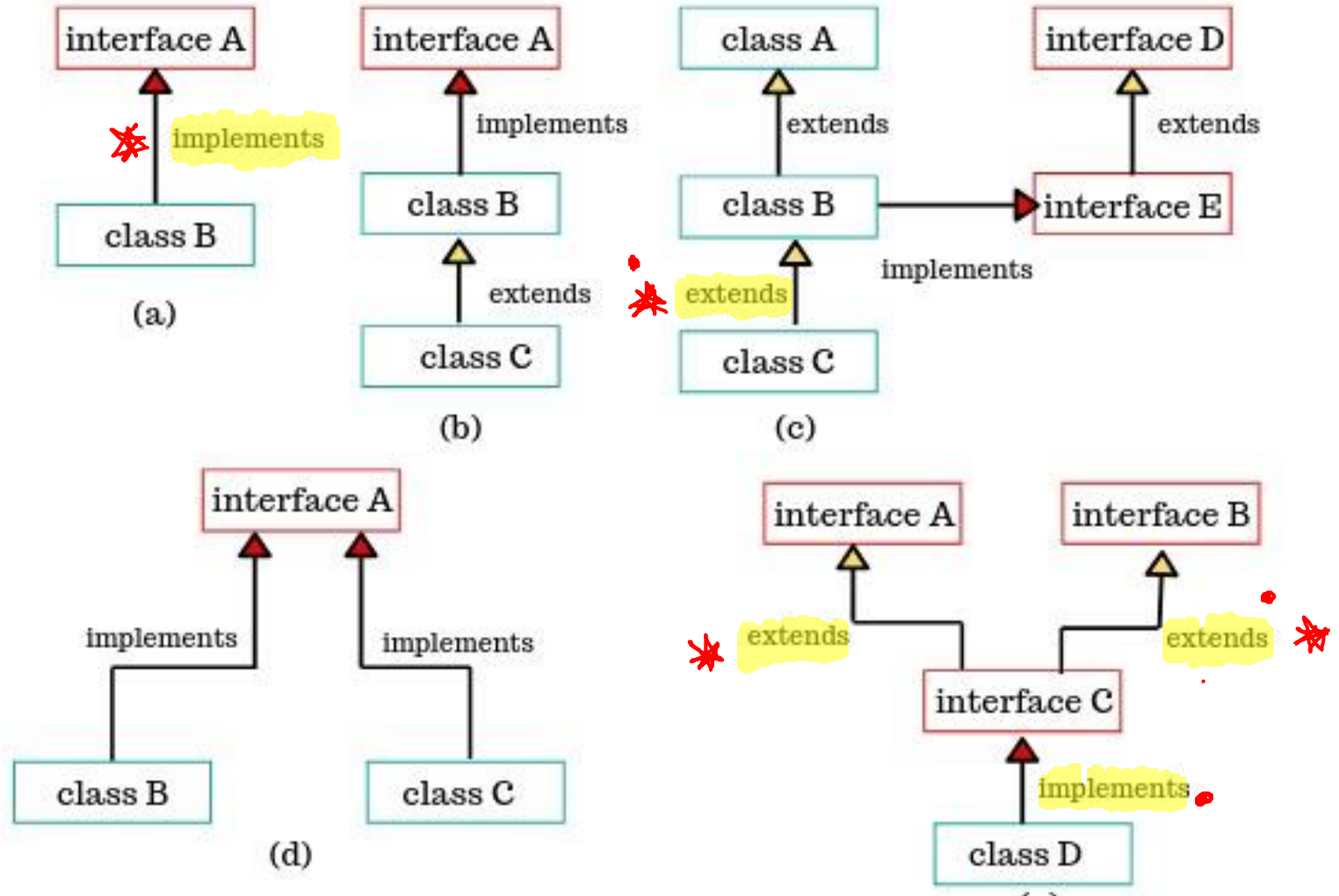
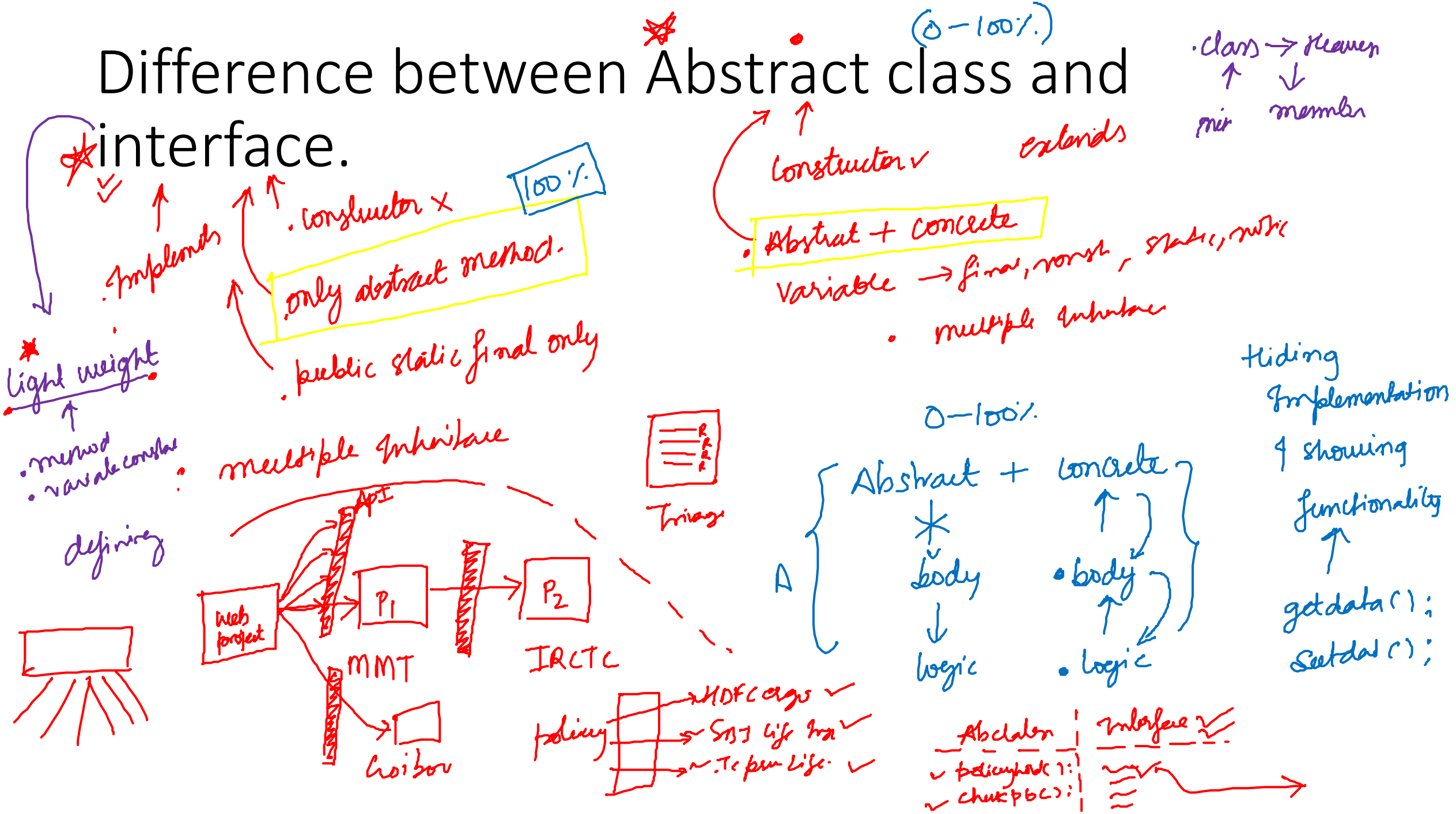


Fig: Various forms of interface implementation

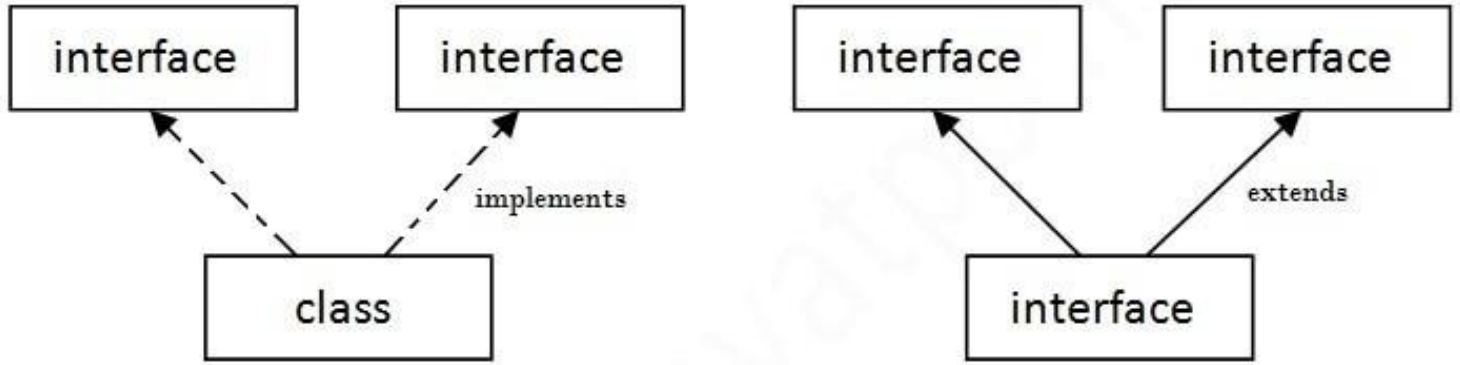
Use of interface coding…

Difference between Abstract class and interface.



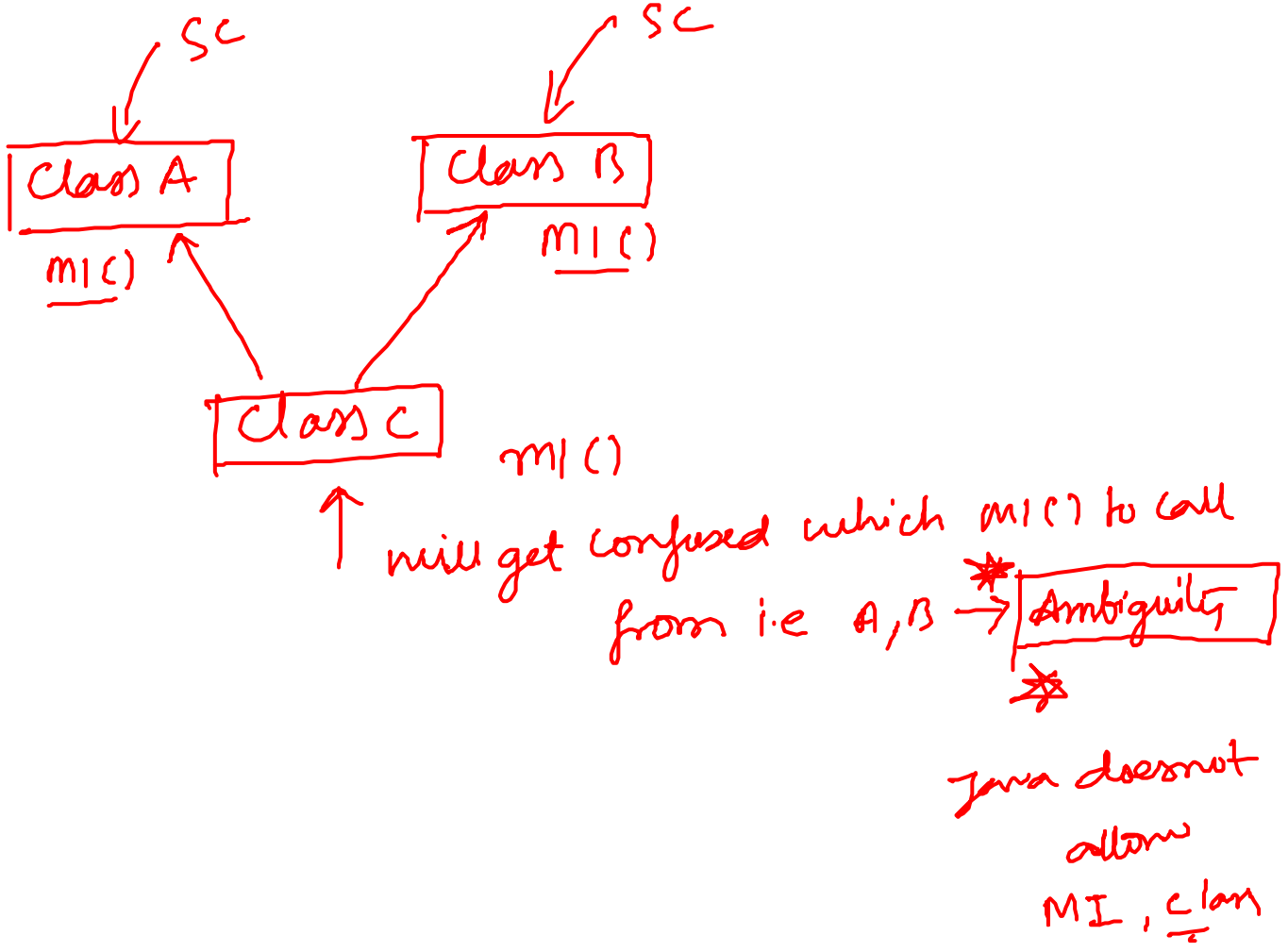
Multiple inheritance in Java by interface

- When a class implements more than one interface, or an interface extends more than one interface, it is called multiple inheritance. Various forms of multiple inheritance are shown in the following figure.



Multiple Inheritance in Java

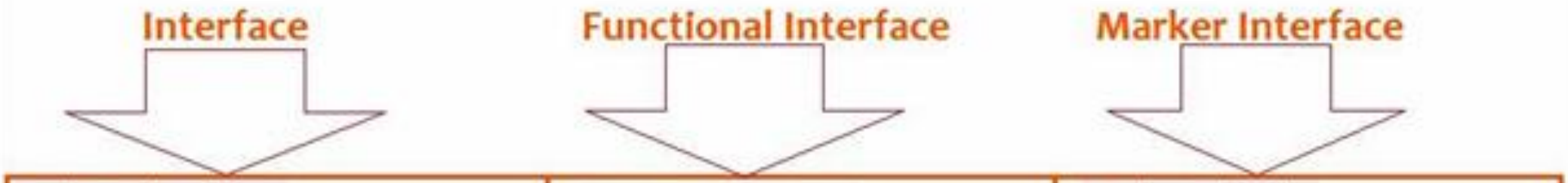
Coding.....


sc
sc
Class A
m1()
Class B
m1()
Class C
m1()
will get confused which m1() to call
from i.e A, B
Ambiguity
Java doesnot
allow
MI, class

In Java, Multiple Inheritance is not supported through Class but it is possible by Interface, why?

- As we have explained in the inheritance chapter, in multiple inheritance, subclasses are derived from multiple superclasses.
- If two superclasses have the same method name then which method is inherited into subclass is the main confusion in multiple inheritance.
- That's why Java does not support multiple inheritance in case of class. But, it is supported through an interface because there is no confusion. This is because its implementation is provided by the implementation class.

Types Of Interfaces In Java




```
package java.lang;

public interface AutoCloseable {

    void close() throws Exception;

}
```

Interface with all abstract methods

```
package java.lang;

@FunctionalInterface
public interface Runnable {

    public abstract void run();

}
```

Interface with One Abstract method

```
package java.io;

public interface Serializable {

}
```

Interface Without any method

Normal Interface: Normal Interface is an interface which has either one or multiple number of abstract methods.

Marker interface: Marker Interface is an interface with no abstract method.

- It is also known as a tagging interface and is used to indicate or inform the JVM that a class implementing this interface will have some special behavior.
- uses, built-in (Serializable, Cloneable, and Remote Interfaces)

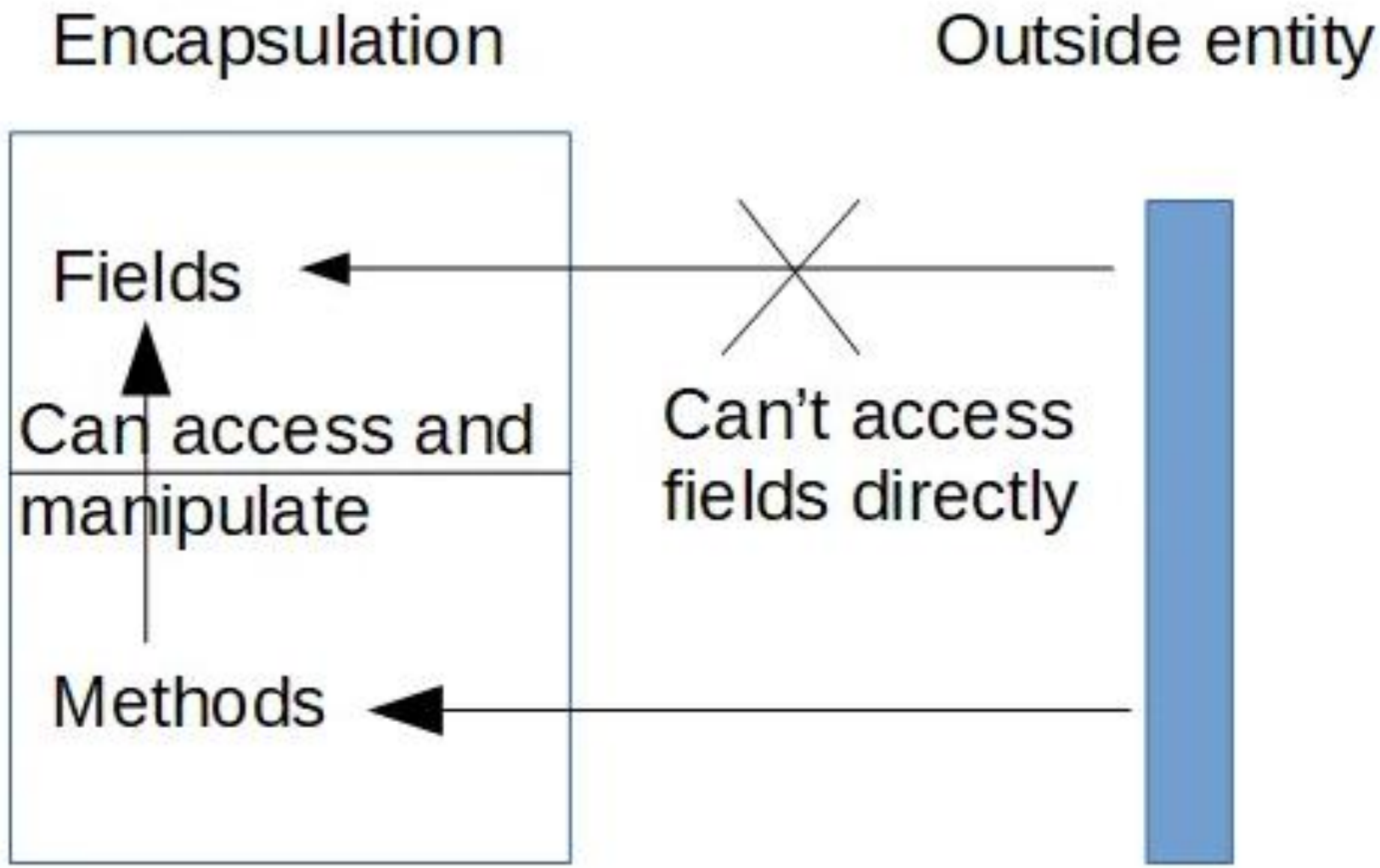
Functional Interface: Functional Interface is an interface which has only one abstract method. Further, it can have any number of static, default methods and even public methods of java.lang.Object class.

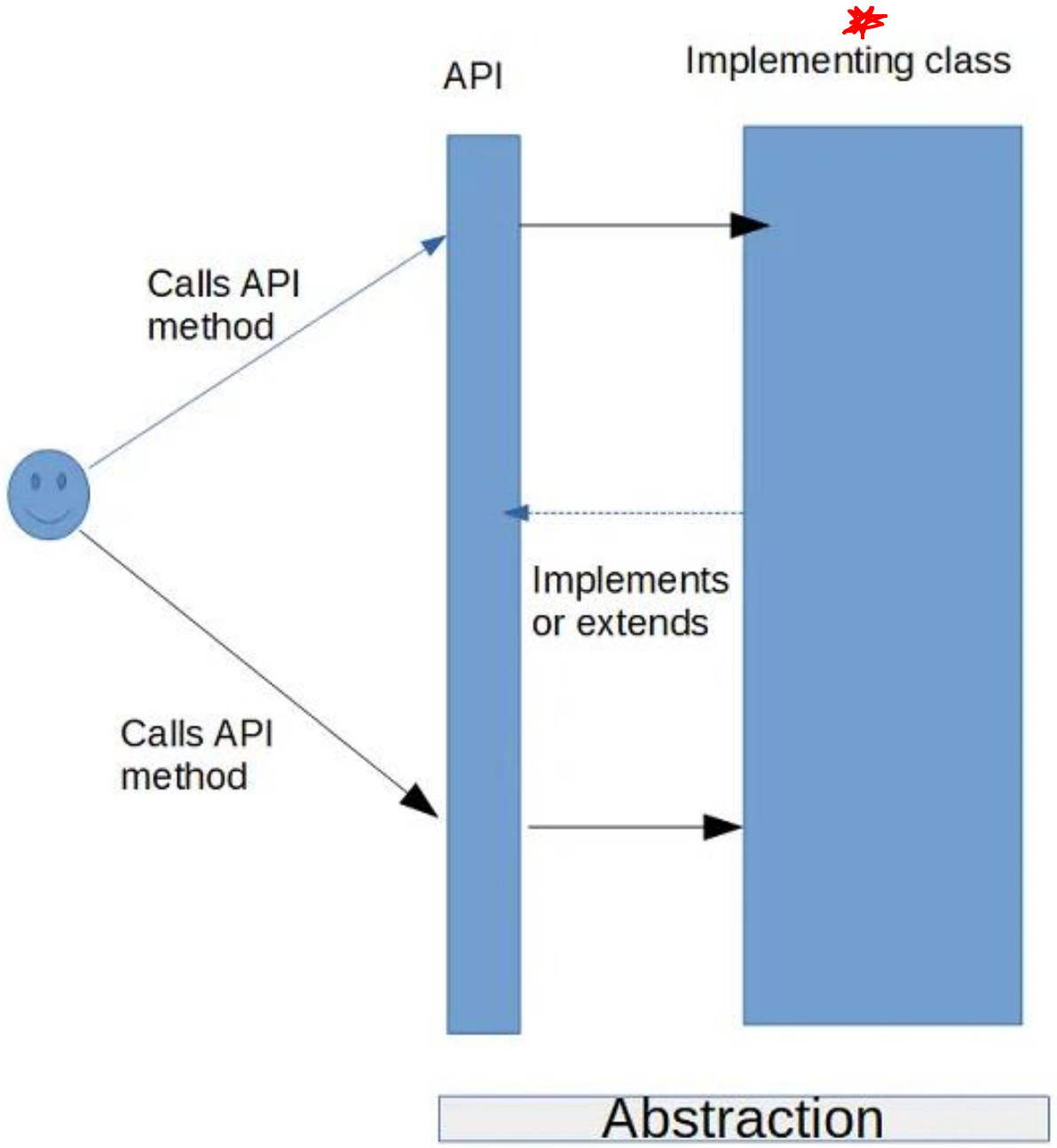
- Runnable: contains only run() method
- Comparable: contains only compareTo() method
- ActionListener: contains only actionPerformed()
- Callable: contains only call() method

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

Difference between Encapsulation and Abstraction.

S.no	Abstraction	Encapsulation
1.	Abstraction is a process of hiding the implementation details and showing only functionality to the user.	Encapsulation is a process of wrapping code and data together into a single unit
2.	Abstraction solves the problem in the Design Level.	Encapsulation solves the problem in the Implementation Level.
3.	Abstraction is implemented by using Interfaces and Abstract Classes.	Encapsulation is implemented by using Access Modifiers.
4.	Abstraction means hiding implementation complexities by using interfaces and abstract class.	Encapsulation means hiding data by using setters and getters.


Encapsulation
Outside entity
Fields
Can access and
manipulate
Methods
Can't access
fields directly


API
Implementing class
Calls API
method
Implements
or extends
Calls API
method
Abstraction

An example of abstraction in Java is-

- An example of abstraction in Java is- Java Database Connectivity (JDBC) API which provides universal data access from the Java programming language. Using the JDBC API, we can access virtually any data source without knowing how the driver for that particular data source is implemented. All we have is an API with a given set of methods.

- BeanFactory and ApplicationContext in spring