

Java Thumb rules

1. If LHS is equal to RHS, then LHS is always a concrete method.

```
A ref=new A();
```

2. If LHS is not equal to RHS, then LHS can be "Interface or concrete or abstract class".

```
B ref=new A();
```

Example:

```
Public Class A
```

```
{  
}
```

```
Public abstract Class B
```

```
{  
}
```

```
Interface Class C
```

```
{  
}
```

```
Class Demo extends A
```

```
{  
}
```

```
Class Demo1 extends B
```

```
{  
}
```

```
Class Demo2 implements C
```

```
{  
}
```

```
A a=new A() // concrete method
```

```
LHS==RHS
```

```
A a=new B()// concrete method
```

```
LHS!=RHS
```

```
B b=new Demo1(); // Abstract class B
```

```
LHS!=RHS
```

```
C c=new Demo2(); // Interface C
```

```
LHS!=RHS
```

3. A class which is declared with "abstract" keyword is an abstract class.

4. A class which is declared without an "abstract" keyword is a concrete class.

5. All final classes are concrete class.

In java class can be declared as a final, final class can be instantiated but cannot be inherited, hence all final classes are concrete class. If a class is an abstract class, it should be inherited to implement, since final class cannot be inherited it cannot be an abstract class.

```
Final abstract class demo
```

```
{  
Demo d=new Demo();
```

```
}
```

Class sample extends demo // error

```
{  
}
```

6. Simply having an interface is of "No use", it should contain "at least one" implementation class.

7. Simply having an abstract class is of "No use", it should contain "at least one" sub class.

8. In java a super class can either be an abstract class or concrete class.

Abstract class A

```
{  
}
```

Class B

```
{  
}
```

Class Demo extends A

```
{  
}
```

Class Demo1 extends B

```
{  
}
```

9. anything in java starting with uppercase letter is either class name or interface name.


10. anything in java starting with lower case letter without parenthesis is a variable.

11. Anything in java ends with parenthesis and starting with lower case is a method.

Class Student


```
{  
  Int studentId;  
  Void studentDetails()  
  {  
  }  
}
```

Scanner
System
String



Classes

next()
println()



Methods

Length → variable

12. In java anything apart from "Primitive data types" are called as "Object reference".

Eg:

Int I;

ABC ref;

- I is a primitive data type
- "Ref" is an object reference variable
- "AB" can be an interface, abstract class or a concrete class.

13. There are two types of interfaces:

Marker interface

Non-marker interface

If a class implements an interface and we are getting implementation error means it must be a non-marker interface and this interface should have one or more abstract methods.

If a class implements an interface and we are not getting any error, then it can be a marker or non-marker interface.

Interface A

{

//Marker

}

Interface B

{

int a;

// Non-marker

}

Interface C

{

Abstract void m1();

}// non-marker

14. If a method/constructor input parameter is other than primitive data type, then it can be "Interface/concrete class / abstract class".

Eg:

Public void(A ref)

{

}

- ref is an object reference variable
- A can be a concrete, abstract or interface

public class InputParameterMain

{

static void m1(**char** ch)

{

System.**out**.println(ch);

}

static void m2(A a)

{

System.**out**.println(a);

}

static void m3(B b)

{

```

        System.out.println(b);
    }
    static void m4(C c)
    {
        System.out.println(c);
    }
    public static void main(String[] args)
    {
        m1('a')
        ;
        m2(new A());
        m3(new Demo2());
        m4(new Demo3());
    }
}

```

15. If method return type is other than primitive type, then it can be interface or abstract or concrete class.

```

Public Xyz method(ABC ref)
{
}

```

Here Xyz can be either an abstract class, concrete class or an interface.

```

public class InputParameterMain
{
    static char m1(char ch)
    {
        return ch;
    }
    static A m2(A a)
    {
        return a;
    }
    static B m3(B b)
    {
        return b;
    }
    static C m4(C c)
    {
        return c;
    }
    public static void main(String[] args)
    {
        char b=m1('a');
        System.out.println(b);
        A a1=m2(new A());
        System.out.println(a1);
        B d=m3(new Demo2());
        System.out.println(d);
        C d1=m4(new Demo3());
        System.out.println(d1);
    }
}

```