* a. equals (b) (method call)

i) Applicable : objects (reference type), not Primitive types.

ii) Purpose : compares the content or logical equality of two objects.

iii) Calls the equals () method defined in the class of object a.

By default (in object class), equals() behaves the same as == (compares memory addresses).

~~most class~~

example:

```
String s1 = new String("Hello");
String s2 = new String("Hello");
boolean s1 = s1.equals(s2);
```

* a == b (Equality

i) both Applicable : both Primitive types and reference types.

ii) Purpose : For Primitive : compares value.

iii) Behavior : checks if values are identical.

checks if both reference Point to the same object in memory.

ex Primitives

```
int x = 5;
    y = 5;
boolean result = (x == y) // true
```

Objects

```
String s1 = new String("Hello");
String s2 = new String("Hello");
boolean s4 = (s1 == s2); // false
```

Java Strings are immutable why?

=) Java Strings are immutable (cannot be changed after creation) for the following key reasons

i) Security: Prevents unauthorized modifications (file Paths, network URLs, database connections)

ii) Thread Safety: Immutable objects are automatically thread-safe

iii) Hashcode caching - Since string is immutable, it's hashcode is can be cached for faster lookups in Hashset, Hashmap, etc.

iv) consistency in collections: Ensures keys in Hashmap or Hashset

eg: example:

```
String s1 = "Hello";
s1.concat("world");
System.out.println(s1);
```