Sub: _____

Sat Sun Mon Tue Wed Thu Fri

Date / /

1. Difference between :

i: a==b

ii: a.equals(b)

Answer

i: a==b

- Checks if booth a and b refer to the same memory location.
- For primitive types (int, char, etc):
  Compares actual values.
- For object (String, Array, etc):
  Compares references (addresses).

for Example:

```
String x = "Hello";
String y = "Hello";
System.out.println(x==y);
String z = new String("Hello");
System.out.println(x==y);
```

Sub:

Date / /

Sat Sun Mon Tue Wed Thu Fri

ii. a.equals(b)

- Compares contents or values inside a and b.
- Used for objects to check if their data is equal.
- Works only if the class has overriden equals() method properly (e.g. String, Integer).

For Example:

```
String x = "Hello";
String z = new String("Hello");
System.out.println(x.equals(z));
```

Sub: _____

Sat Sun Mon Tue Wed Thu Fri

Date / /

2/ Why Java String immutable ?,

Ans:

Once a string object is created, its value cannot be changed. if you try to change it a new string object is created instead.

Reasons for immutability:

i. • strings are used in network connections, file paths, database URLs, usernames, passwords.
• if they were mutable, hackers could change the value after creation, causing security risks.

ii. • Java uses a string pool (interning) to save memory.
• Since strings don't change, multiple references can point to the same string safely.

```
String a = "Hello";
String b = "Hello";
```

iii. Thread safety
Immutable objects are automatically thread safe because their value cannot be changed by any thread after creation.

Sub: _____

Sat Sun Mon Tue Wed Thu Fri

Date / /

iv. Caching and performance:

- Hashcode of a string is cached (used in HashMap keys)
- If String was mutable, hashcode would change and cause data inconsistency in collections.

v. Reliability in Operations:

String immutability makes functions and libraries more predictable and reliable as String data remains unchanged.

```
String s = "Hello";
s = s + "word";
```