**Day-1 Java Assignment**

**1. Problem Description:**

What is the difference between equals() and == in Java?

Tell the output of the given code.

public class Dog {}

public class Test {

public static void main(String[] args) {

int n1 = 10;

int n2 = 10;

System.out.println(n1 == n2);

boolean b1 = true;

boolean b2 = false;

System.out.println(b1 == b2);

Dog dog1 = new Dog();

Dog dog2 = new Dog();

System.out.println(dog1 == dog2);

System.out.println(dog1.equals(dog2));

String s1 = "Hello! World!";

String s2 = "Hello! World!";

String s3 = new String("Hello! World!");

System.out.println(s1 == s2);

System.out.println(s1 == s3);

System.out.println(s1.equals(s2));

System.out.println(s1.equals(s3));

}

}

**2. My Solution:**

**== is an operator.**

* If we are using the == operator to compare primitive data type values, then internally actual values in the memory will be compared to produce the result.
* If we are using the == operator to compare class objects, then internally, actual memory addresses will be compared to produce the result.

**equals() is a method present in Java.lang.Object:**

* If we are comparing string objects, then internally, first memory addresses will be checked; if the memory addresses of two string objects are the same, the equals method will return true; otherwise, it will intern check the contents, and if the contents are the same, it will return true; otherwise, it will return false. This specific function definition is implemented in the overridden equals (object object) method in the string class, and the string class is a child of the java.lang.Object. All the classes in Java are by default child classes of java.lang.Object class.
* If we are comparing other than string objects and those respective classes did not override the equals (Object obj) method in the parent Object class, then actual Java.lang the Object class equals object method will be executed.

**Question:**

**What java.lang.Object class equals(Object object) method will do?**

* It will just compare the memory addresses, if the memory addresses are the same, then the method will return true, otherwise false.

**Answers to given code:**

public class Dog {}

public class Test {

public static void main(String[] args) {

int n1 = 10;

int n2 = 10;

System.out.println(n1 == n2); //True, we are checking primitive type values using the "==" operator, so internally it will compare the values of the variables.

boolean b1 = true;

boolean b2 = false;

System.out.println(b1 == b2); //False, internally, the == operator checks if the values of the Boolean variables are equal.

Dog dog1 = new Dog();

Dog dog2 = new Dog();

System.out.println(dog1 == dog2); //False, using == operator, we are comparing the memory addresses or references of the objects, not their actual content.

System.out.println(dog1.equals(dog2)); //False, it compares object references (memory addresses) by default. Since both abject has different references the resultant is False.

String s1 = "Hello! World!";

String s2 = "Hello! World!";

String s3 = new String("Hello! World!");

System.out.println(s1 == s2); //True, both s1 and s2 refer to the same string literal in the string pool, which is why the result is True

System.out.println(s1 == s3); //False, In this example memory reference is different, but the value is those diff. memory is same.

System.out.println(s1.equals(s2)); //True, the equals() method compares the content of the strings s1 and s2. Since both s1 and s2 are string literals with the same content "Hello! World!", the comparison evaluating true.

System.out.println(s1.equals(s3)); //True, here equals() method compares the content of the strings s1 and s3. Despite being created differently (one is a string literal, the other is a new String object), both s1 and s3 contain the same sequence of characters "Hello! World!".

}

}