**জাভা তে String**

জাভা তে স্ট্রিং দুইভাবে ডিক্লেয়ার করা যায়।

//Way 1String str=“Hello”;  
//Way 2String str2=new String(“Hello”);

প্রথমত, আমরা ভেরিয়েবল এর মতো করে String ডিক্লেয়ার করতে পারি। আর দ্বিতীয়ত, new এর মাধ্যমে আমরা String Class এর Object কে Instantiate করতে পারি। উপরের Code Snippet এ দুইটি String দুইভাবে ডিক্লেয়ার করা হয়েছে। এবং প্রত্যেকের মধ্যে ভেল্যু হিসেবে দেয়া আছে “Hello”।

নিয়মানুযায়ী এই দুটি অবজেক্ট সমান কি না তা চেক করতে গেলে তা সমান দেখানোর কথা। এইখান থেকে শুরু হয়ে যায় স্ট্রিং এর যত বিভ্রান্তি, যা কিছুক্ষনের মধ্যেই দূর হয়ে যাবে বলে আশা করছি।

নিচের Code টা একটু খেয়াল করিঃ

public class StringDemo {  
 public static void main(String []args) {  
 String str1="Hello";  
 String str2=new String("Hello");  
   
 //condition1  
 if(str1==str2) {  
 System.out.println("Both objects carry the same reference");  
 }  
 else {  
 System.out.println("These two objects carry different references");  
 }  
 //condition2  
 if(str1.equals(str2)) {  
 System.out.println("The values of both strings are same");  
 }  
 else {  
 System.out.println("The values are different");  
 }  
 }  
}

এখানে প্রথম if কন্ডিশনে দুইটি স্ট্রিং অবজেক্ট কে == দিয়ে চেক করা হয়েছে।

দ্বিতীয় if কন্ডিশনে দুইটি স্ট্রিং অবজেক্ট কে equals() মেথড দিয়ে চেক করা হয়েছে। এই মেথডটি String ক্লাস এর একটি মেথড, যা আর্গুমেন্ট হিসেবে আরেকটি স্ট্রিং অবজক্টকে নেয় এবং চেক করে যে অবজেক্টটির মেম্বার মেথড হিসেবে একে কল করা হয়েছে, তা আর্গুমেন্ট এর সমান কি না। str1.equals(str2) লিখে এখানে তা-ই বোঝানো হয়েছে।অর্থাৎ এটি দুইটি স্ট্রিং এর Value সমান কি না তা চেক করে।

কোডটি রান করলে প্রথম if কন্ডিশনটি false আউটপুট দেবে, দ্বিতীয়টি দেবে true।

কেন!!!!!!!!!!

**String Pool**

জাভা তে দুই ধরনের মেমোরি রয়েছে। এরা হচ্ছে Stack Memory এবং Heap Memory. সাধারনত ফাংশন কল এবং লোকাল ভেরিয়েবল,প্যারামিটার এগুলো ব্যবহার করে Stack Memory । আর অবজেক্ট ব্যবহার করে থাকে Heap memory। স্ট্রিং নিয়ে কাজ করার সময় আমরা String Class এর অবজেক্টই তৈরী করে থাকি। Heap Memory তে আরেকটি সেকশন আছে, যার নাম String Pool। যখন আমরা ভেরিয়েবল এর মতো “Hello” কে “=” অপারেটর দ্বারা str1 এর মধ্যে অ্যাসাইন করছি, তখন String Pool এর মধ্যে একটি জায়গা তৈরী হয় এবং “Hello” সেই জায়গায় চলে যায়। এই “Hello” স্ট্রিং এর Reference বহন করে str1 দ্বিতীয় অবজেক্টটি new দিয়ে তৈরী করাতে তার জন্য স্বাভাবিকভাবে Heap Memory তে (Pool এর বাইরে) একটি মেমোরি বরাদ্দ হয়েছে। এবং এর Reference বহন করছে str2। প্রথম if কন্ডিশনে == অপারেটর দ্বারা আসলে এদের Value চেক করা হয় না, চেক করা হয় Reference । তাই এটি false হয়ে else কন্ডিশনে চলে যায়।

অপরদিকে, equals() মেথডটি দিয়ে চেক করা হচ্ছে এদের Value. তাই এটি true দেখাবে।

এবার আরেকটা Scenario দেখা যাক

public class StringDemo {  
 public static void main(String []args) {  
 String str1="Hello";  
 String str2="Hello";  
   
 if(str1==str2) {  
 System.out.println("Both objects carry the same reference");  
 }  
 else {  
 System.out.println("These two objects carry different references");  
 }  
   
 if(str1.equals(str2)) {  
 System.out.println("The values of both strings are same");  
 }  
 else {  
 System.out.println("The values are different");  
 }  
 }  
}

আচ্ছা এবার দুইটি অবজেক্ট একই পদ্ধতিতে ডিক্লেয়ার করলাম। যখনই আমি (str1==str2) লিখে চেক করতে যাবো, তা true দেখাবে। কারন, str2 যখন ডিক্লেয়ার হলো, তখন আর নতুন করে String Pool এ “Hello” কে পাঠানো হয় নি কারন “Hello” সেখানে আগে থেকেই ছিলো। সেক্ষেত্রে Pool এ পূর্বের “Hello” এর Reference টা str2 বহন করবে, আর আমরা এখন জানি == অপারেটর দিয়ে Reference চেক করা হয়, Value নয়। তাই এরা সমান।

আরেকটি Scenario লক্ষ করিঃ

public class StringDemo {  
 public static void main(String []args) {  
 String str1="Hello";  
 String S="He";  
 String str2=S+"llo";  
   
 if(str1==str2) {  
 System.out.println("Both objects carry the same reference");  
 }  
 else {  
 System.out.println("These two objects carry different references");  
 }  
   
 if(str1.equals(str2)) {  
 System.out.println("The values of both strings are same");  
 }  
 else {  
 System.out.println("The values are different");  
 }  
 }  
}

এখানে S নামের আরেকটি স্ট্রিং কে str2 এর সাথে concat করা হয়েছে। এখন আউটপুট কি আসার কথা?

একটি অবজেক্ট কে যখন আরেকটি অবজেক্ট এর সাথে concat করা হয় বা জোড়া লাগানো হয়, তখন Object Reference পালটে যায়। তাহলে এক্ষেত্রে == অপারেটর দিয়ে চেক করতে গেলে তা অবশ্যই false দেখাবে।

https://miro.medium.com/max/60/1*T5RpDSWA-48vIpcEXjXRDA.png?q=20

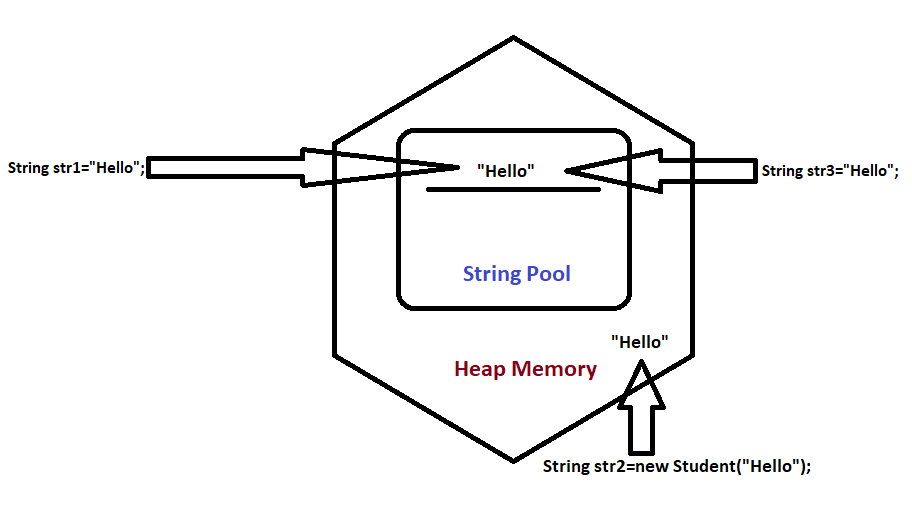


Diagram of String Pool

**String vs. StringBuffer**

String নামের যে ক্লাসটি রয়েছে, তাকে বলা হয় Immutable, অর্থাৎ, যাকে Modify করা যায় না। String ক্লাস এর কোন অবজেক্ট একবার তৈরী হয়ে গেলে পরে তার ভিতরের কিছু বদলানো যায় না। বদলাতে যদি হয়ই, তবে পুরো অবজেক্ট কে আবার Overwrite করতে হবে।

public class StringDemo {  
 public static void main(String []args) {  
 String str="Hello";  
 str.concat("World");  
 System.out.println(str);  
 }  
}

concat() নাম এর যে ফাংশনটি ব্যবহার করা হয়েছে, তা String ক্লাস এর ফাংশন। এটি একটি নতুন স্ট্রিং কে এর অবজেক্ট এর স্ট্রিংটির সাথে জোড়া দেয়, কিন্তু Permanently তা করে না। কোডটি রান করলে আউটপুট আসবে ঃ

Hello

যদি আমি আমার আশানুরূপ আউটপুট দেখতে চাই, তাহলে লিখতে হবেঃ

System.out.println(str.concat("World"));

concat() ফাংশনটি আমার আউটপুটকে রিটার্ন করে, যা আমি println() মেথড দিয়ে দেখালাম।আর যদি আমি পুরোপুরিভাবে জোড়া লাগাতে চাই, তাহলে কোডটি হবে এমনঃ

public class StringDemo {  
 public static void main(String []args) {  
 String str="Hello";  
 str=str.concat("World");  
 System.out.println(str);  
 }  
}

এখানে নতুন করে আবার str এর মধ্যে “Hello” এবং “World” কে Concat করে রাখা হয়েছে।

String ব্যবহার করা হয়ে থাকে এমন Character Sequence কে Manipulate করার জন্য, যা অপরিবর্তিত, অর্থাৎ Immutable । পরিবর্তনশীল স্ট্রিং নিয়ে কাজ করলে ব্যবহার করা হয় StringBuffer । এটি Mutable। নিচের কোডটি দেখলেই বোঝা যাবে।

public class StringDemo {  
 public static void main(String []args) {  
 StringBuffer sbf=new StringBuffer("Hello");  
 sbf.append("World");  
 System.out.println(sbf);  
 }  
}

append() ফাংশনটি আসলেই অন্য একটি স্ট্রিং কে অবজেক্ট স্ট্রিং এর সাথে জোড়া লাগায়। এই কোডের আউটপুট আসবেঃ

HelloWorld

আরেকটি Mutable ক্লাস এর নাম হচ্ছে StringBuilder । এর কাজ করার পদ্ধতি StringBuffer এর মতো হলেও একটু আলাদা। এই দুটি ক্লাস Concatenation এর বেলায় String এর তুলনায় অনেক বেশি দ্রুত।কারন আমরা যা দেখলাম, concat() মেথডটি ব্যবহার করে আবার str এর মধ্যে এসাইন করতে হচ্ছে, তার মানে প্রত্যেকবার নতুন অবজেক্ট তৈরী হচ্ছে। কিন্তু StringBuffer এর বেলায় আমরা আগের অবজেক্ট এর সাথেই নতুন স্ট্রিং যোগ করে দিচ্ছি। এখানেই আসলে মূলত String এর সাথে StringBuffer ও StringBuilder এর তফাৎ।

**Console থেকে String ইউজার ইনপুট নেওয়া**

সবার আগেই আমার Scanner ক্লাস এর একটি অবজেক্ট লাগবে, যা দ্বারা আমি ইনপুট নেব।

Scanner scan=new Scanner(System.in);

তার জন্য আগে সবার উপরে util প্যাকেজ থেকে Scanner ক্লাসটিকে import করে নেব।

import java.util.Scanner;

এরপরে আমার ডিক্লেয়ার করা স্ট্রিং এর মধ্যে ইনপুটকে রাখার ব্যবস্থা করতে হবে।

String myString=scan.nextLine();

এখানে scan হচ্ছে Scanner ক্লাস এর object, যা দ্বারা আমি nextLine() ফাংশন কল করলাম। একে কল করা মাত্রই সে ইউজার ইনপুট চাইবে। ইনপুট নেওয়া হয়ে গেলে সে ইনপুট স্ট্রিং কে আমার ডিক্লেয়ার করা myString অবজেক্ট এ এসাইন করে দিবে। তাহলে পুরো কোডটি দেখতে এমন হবেঃ

import java.util.Scanner;  
public class StringDemo {  
 public static void main(String []args) {  
 Scanner scan=new Scanner(System.in);  
   
 String myString;  
   
 myString=scan.nextLine();  
   
 System.out.println("You have entered : "+myString);  
 }  
}