**Data Types:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Type** | **Wrapper Class** | **Byte** | **Bits** | **Max - Min** |
| byte | Byte | 1 | 8 | Stores whole numbers from -128 to 127 |
| short | Short | 2 | 16 | Stores whole numbers from -32,768 to 32,767 |
| int | Integer | 4 | 32 | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | Long | 8 | 64 | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
|  |  |  |  |  |
| float | Float | 4 | 32 | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | Double | 8 | 64 | Stores fractional numbers. Sufficient for storing 15 decimal digits |
|  |  |  |  |  |
| char | Character | 2 | 16 | Stores a single character/letter or ASCII values |
| String | String | Depend on value (Each character has 2 Bytes) | Depend on value (Each character has 8 Bits) |  |
|  |  |  |  |  |
| bool | Boolean | 1 bit | 8 | Stores true or false values |

**Strings:**

String manipulation is one of the most common activities in programming, String in Java are unique due to the way they get store in java.

String is an object of Java.lang.String, we can instantiate the string using String Object.

String S = new String (); // empty String

String S = new String(“hello”); // “hello” String literal

String literal is a String object; Internally String literal passe as String Object.

Alternate way to create String is:

Char [] charArray = { ‘H’ , ‘E’ , ‘L’, ‘L’, ‘O’};

String s = new String (charArray);

String s = “hello”; // String literal (Variable s is still an object reference, do not take it as literal)

Recommended way to create String: **String s = “hello”;**

String Class use character array to store the text. String in Java basically sequence of Unicode character. Java uses **UTF-16** for character representation.

String in java is immutable, once the value is created it cannot be change. String is very special in java.

+ Operator is used for concatenation of multiple the String;

String s = “hello” + “java”; // “hello world”

**String pool ~**

All string created using String literal are saved in special are of memory called String pool. This does not happen String created using new keyword. They do get store in string pool. Main advantage of String pool is that if two string are created using same literal they both would represent by the same string object in the string pool. But if two String are created using new keyword with same text, then they both are still represent the different objects. That’s why it’s highly recommended to create String using String word not with new keyword. It will save the memory and time as well.

**How Strings are stored in JVM, Using String Literal vs Using new.**

**String s1 = “Hello”; String s4 = new String (“Hello”);   
String s1 = “Hello”;  
String s3 = s1;**

**s3**

**s1**

**s**2

Hello

**heap**

String pool

**Explanation:**

If we have two String variable by the same literals they will share same memory, if string literal have same content, they share same storage in string pool. In upper case s1 and s2 will refer the same storage. These two statements can appear anywhere in application they maybe in totally different classes or maybe in totally different packages together. As long as long as both statements are executed as part of JVM only single reference will be created and s1 and s will be refence it. Net if s3 String variable assign with s1 that will also would refer the same memory reference in string pool.

Next if a variable is created by using new statement; also initialized text with “Hello”, this would create the string object outside of the string pool just like regular any java object and s4 would reference it however argument to this variable is string literal JVM will still try to create a string object of that literal in string pool, since we have already string literal in String pool a reference will be passed to string constructor. But if a string literal correspond to hello will not be present in String pool JVM would have created a new string object in string pool and would have pass the reference to that string constructor. To go on String, we need String literal. If we have statement s5 with same literal “Hello”, this would also create a Sting object outside of the String pool, so s4 and s5 are referring different object even though they have the same String literal. There is no sharing of storage between s4, s4 as in term of s1, s2 and s3.

When we do object reference comparison s1 vs s2 we will get True value on other when we compare s4 with s5 we will get result as False. Main advantage of using String literal is saving memory.

**String s1 = “Hello”; String s4 = new String (“Hello”);   
String s1 = “Hello”; String s5 = new String (“Hello”);  
String s3 = s1;**

**s4**

**s5**

**s3**

**s1**

**s**2

Hello

String pool

**heap**

**s1==s2? True**

**s4==s5? False**

If we have only  **String s = new String(“Hello”);**

Statement than JVM encounter String literal for the first time and a corresponding “Hello” object is created on String pool, later a reference is passed outside of String to heap to construct a new string object, in reality in this case two String object are being created. But if “Hello” literal was already in String pool than it would have created only one object outside of the String pool.

String pool stored a copy of each element in the form of String Object. There is only one copy of String pool. This whole process called String interning.