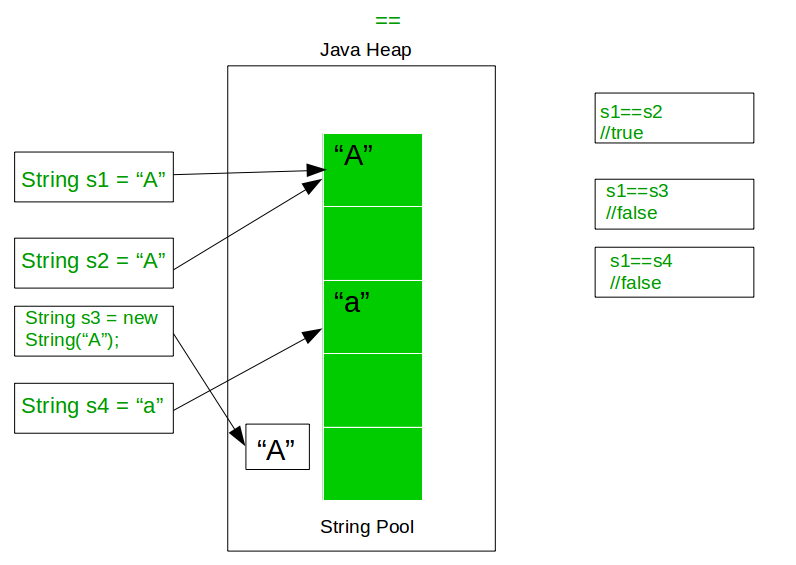
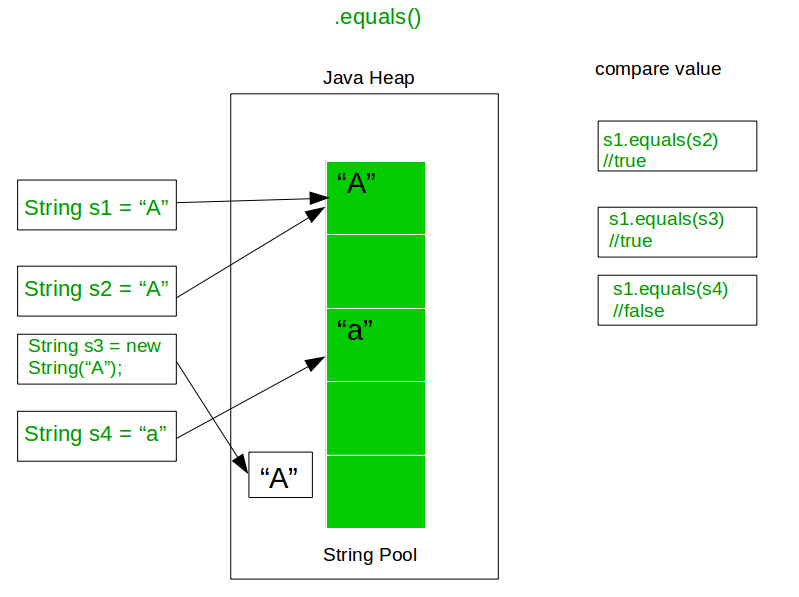
There are many ways to compare two Strings in Java:

* Using == operator//boolean
* Using equals() method//boolean
* Using compareTo() method//return unicode
* Using compareToIgnoreCase() method//boolean
* Using compare() method//custom

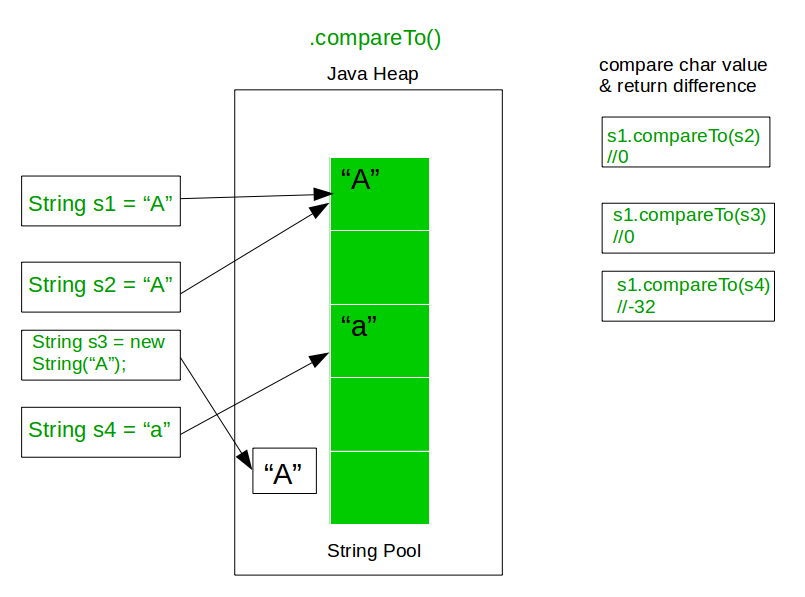
**Method 1: using == operator(compare reference )**



**Method 2: Using equals()**



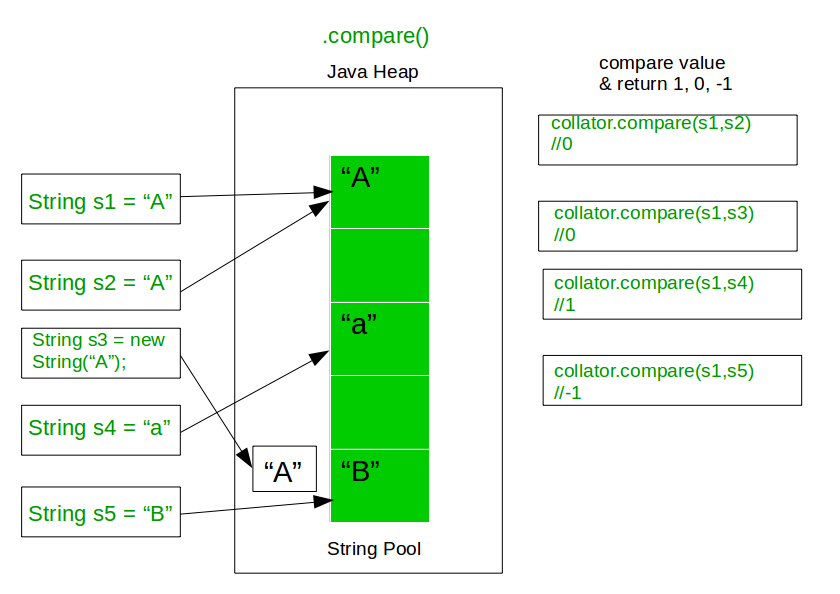
**Method 3: Using compareTo() method(**based on the Unicode value)



**Method 4:**equalsIgnoreCase(String anotherString)//return boolen T or F

**Method 5: Using compare() method(based on 0,1,-1)//according you thinking**

**Collator class** which is in **java.text** package. The one most important feature of **Collator class** is the ability to define our **own custom comparison rules**.



**String important method:**

1**. String[] split(String regex):** It returns an array of strings after splitting an input String based on the delimiting regular expression.

1. String str = new String("28/12/2013");

String array1[]= str.split("/");

for (String temp: array1){

System.out.println(temp);//28 12 2013

}

**Example: word as a regular expression in Java String split method**

2. String str = "helloxyzhixyzbye";

String[] arr = str.split("xyz");

for (String s : arr)

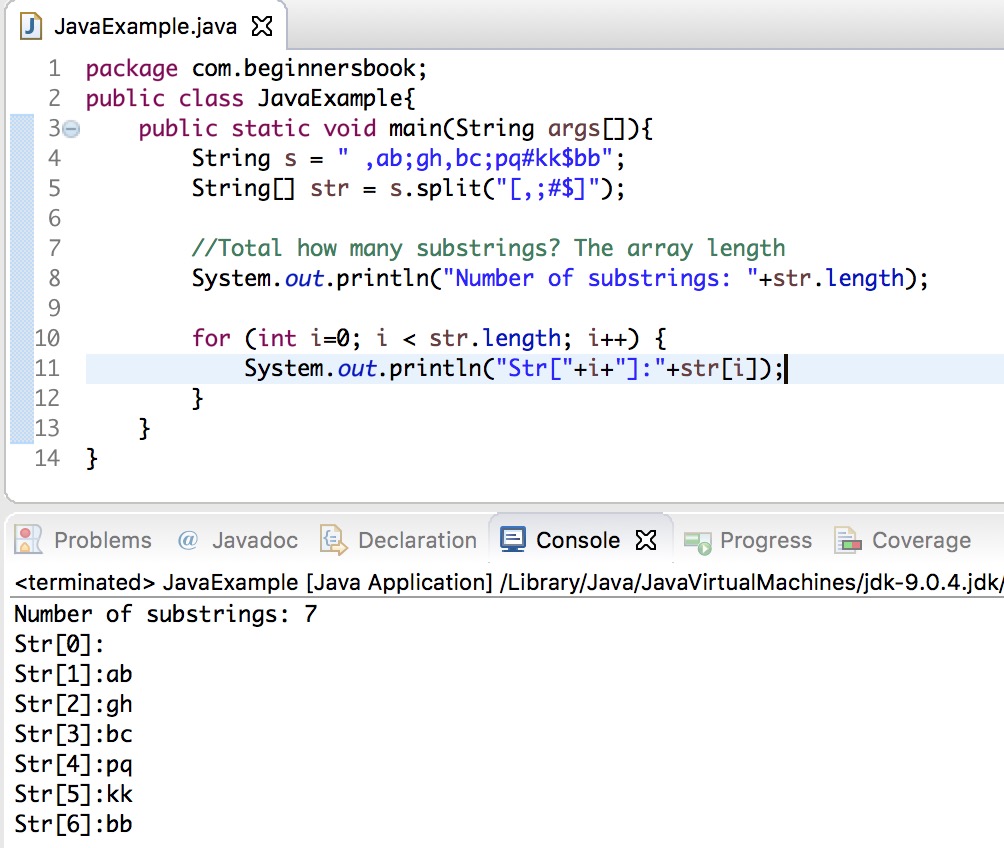
System.out.println(s);//hello hi bye

**Example: splitting string based on whitespace**

String str = "My name is Chaitanya";

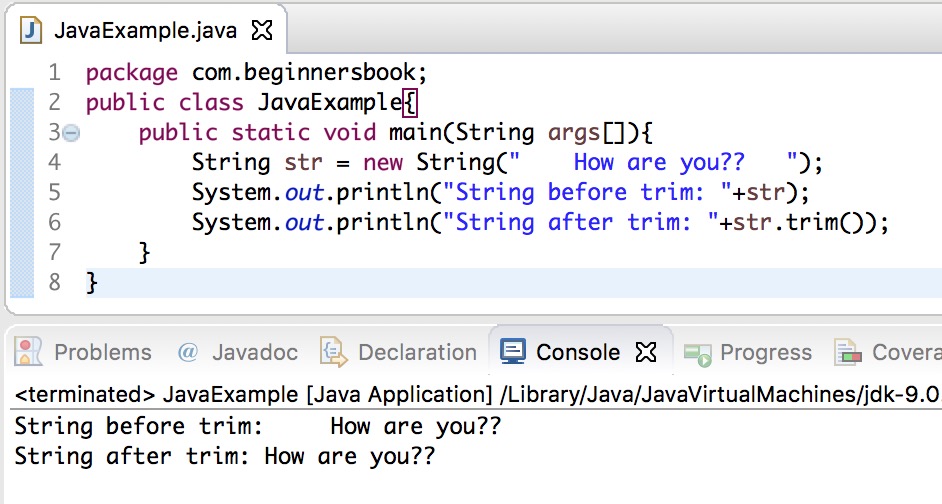
String[] arr = str.split(" ");

4. In this example we are splitting input string based on multiple special characters.

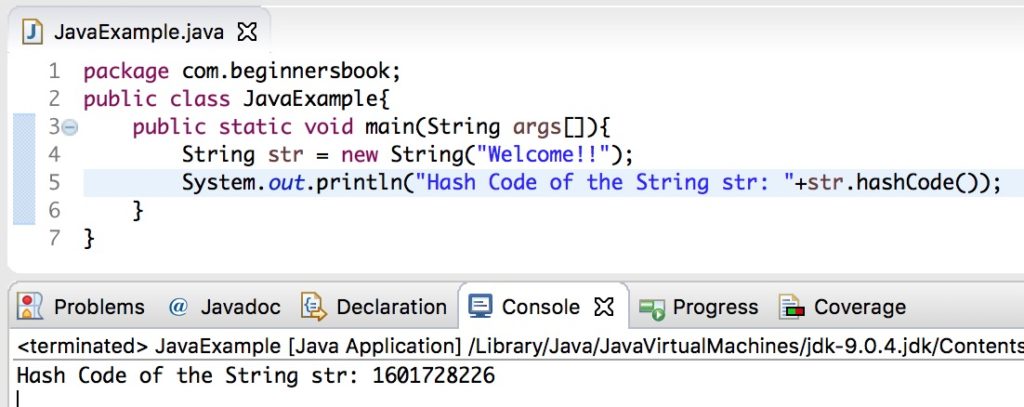


**2. Java String trim() and hashCode()**

**Trim():**It returns a String after removing **leading and trailing white spaces** from the input String.



2.**hashCode**():the hash code of this value using the hashCode() method.



3. **Java String valueOf():** method returns the String representation of the boolean, char, char array, int, long, float and double arguments.

int i = 10; //int value

float f = 10.10f; //float value

long l = 111L; //long value

double d = 2222.22; //double value

char ch = 'A'; //char value

char array[] = {'a', 'b', 'c'}; //char array

//converting int to String

String str1 = String.valueOf(i);

//converting float to String

String str2 = String.valueOf(f);

//converting long to String

String str3 = String.valueOf(l);

//converting double to String

String str4 = String.valueOf(d);

//converting char to String

String str5 = String.valueOf(ch);

//converting char array to String

String str6 = String.valueOf(array);

System.out.println(str1);

System.out.println(str2);

System.out.println(str3);

System.out.println(str4);

System.out.println(str5);

System.out.println(str6);

**Output:**

10

10.1

111

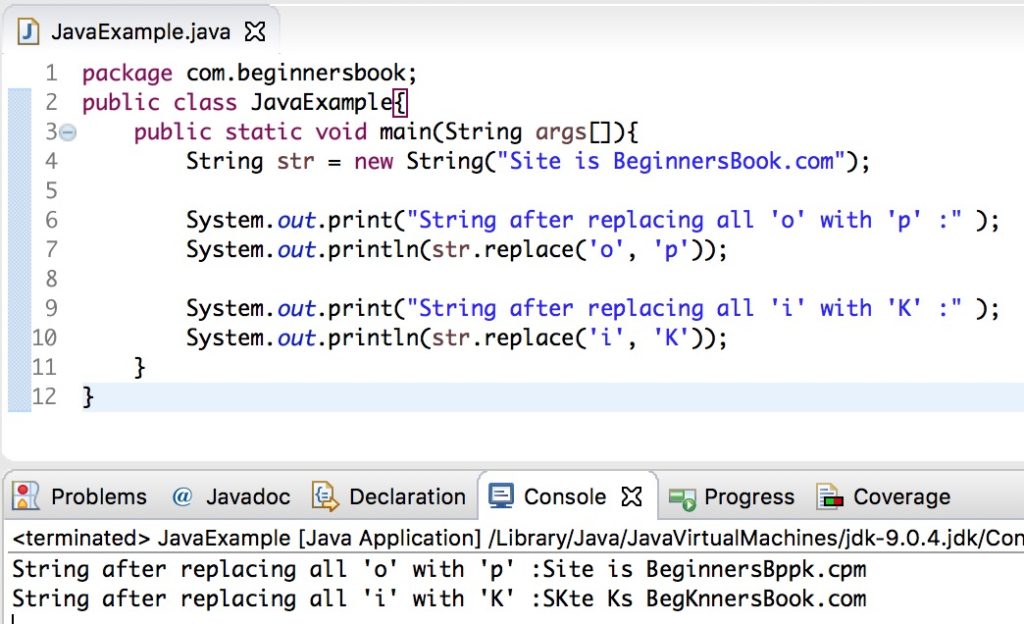
2222.22

A

abc

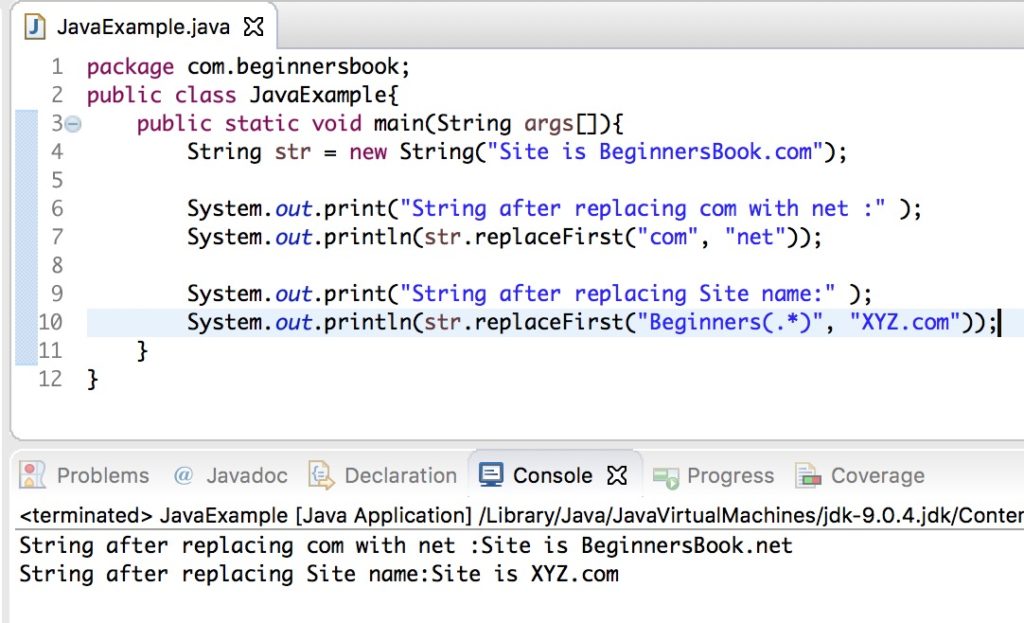
In this tutorial we will discuss replace(), replaceFirst()and replaceAll() methods.

1. **replace():**



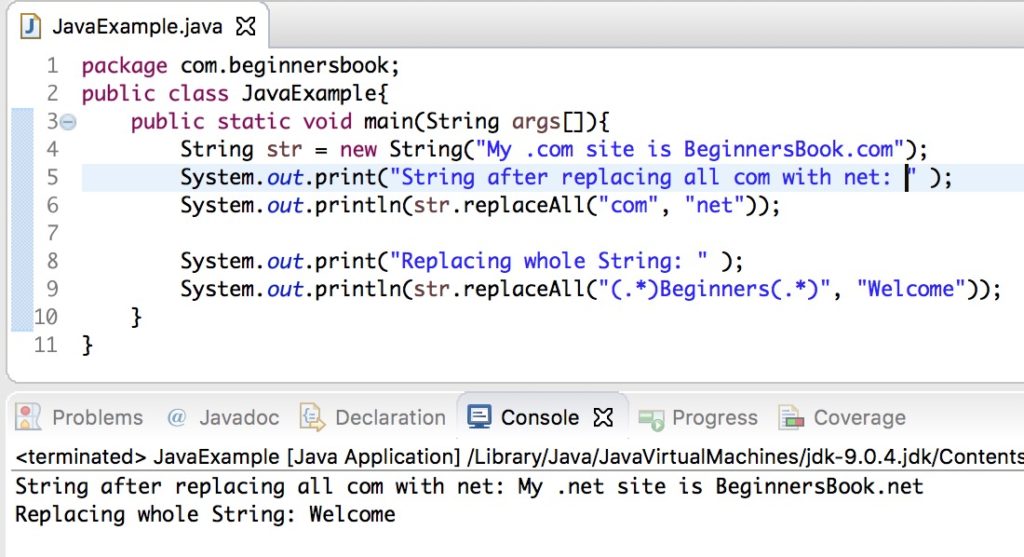
**2. replaceFirst():**

The **difference between replaceFirst() and replaceAll() method** is that the replaceFirst() replaces the first occurrence while replaceAll() replaces all the occurrences.



3. replaceAll(): method to replace all the occurrences of a given substring with the new string.

The **difference between replace() and replaceAll() method** is that the replace() method replaces all the occurrences of **old char** with **new char** while replaceAll() method replaces all the occurrences of **old string** with the **new string.**



**Java String intern()** :method is used for getting the string from the memory if it is already present. This method ensures that all the **same strings** **share the same memory.** For example, creating a string “hello” 10 times using intern() method would ensure that there will be only one instance of “Hello” in the memory and all the 10 references point to the same instance.

public class Example{

public static void main(String args[]){

String str1 = "beginnersbook";

/\* The Java String intern() method searches the string "beginnersbook"

\* in the memory pool and returns the reference of it.

\*/

String str2 = new String("beginnersbook").intern();

//prints true

System.out.println("str1==str2: "+(str1==str2));

}

}

Output:

str1==str2: true

**String Program:**

**String s1="Akash";**

**String s2="Thakare";**

**int x=10;**

**int y=20;**

**Program:**

**System.out.println(s1.substring(1,4));//kas**

**System.out.println(s1=s2);//Thakare**

**System.out.println(s1.indexOf('z'));// -1**

**Program:**

**/\*replace() replace all old character with new Character**

**replaceAll() replace all ols string with new String\*/**

**String s3="Corcock".replace('c', 'd');//replace all character**

**System.out.println(s3);//Cordodk**

**String s4="A1B2C3";**

**String c[]=s4.split("\\d");// "\\s" for space and "\\d" for digit**

**System.out.println(Arrays.toString(c));//[A, B, C]**

**String s5="";**

**System.out.println(s5.length());//0**

**String s6=null;**

**//System.out.println(s6.length());//null pointer exception**

**String s7=new String("akash");**

**StringBuilder s8=new StringBuilder("akash");**

**StringBuffer s9=new StringBuffer("akash");**

**System.out.println(s8.equals(s7));//false**

**System.out.println(s8.equals(s9));//false**

**//System.out.println(s8==s9); C.T error**

**Program:**

**System.out.println(""=="");//in double "" only use =="" it will print true**

**System.out.println("a==A");//print as it is i.e a==A**

**Program:**

**int i=5;**

**i- =3+7;// is same as x = x - (3 +7) first 3+7=10 then 5-10=-5**

**System.out.println(i);//-5**

**Why it is best practice to write any expression like:**

**Int I=5;**

**I\*=5;**

**Instead of i=i\*5;**

**Bz, see example**

**byte b1=10;**

**//b1=(byte)b1+5; OR b1=b1+5//both show C.T ERRor**

**So compiler complain about Type mismatch:**

**So write like this:**

**b1+=5;**

**If the left hand operand is not a String then + operator treat as plus**

**BUT if the left hand operand is a String then + perform String concatenation.**

**int p=10,q=20,r=30;**

**System.out.println("Value is "+p+q);//Value is 1020**

**System.out.println(p+q+r);//60**

**System.out.println("String"+q+r);//String2030**

**System.out.println(q+r+"String");//50String**

**/\*All wrapper classes can be use as variable name\*/**

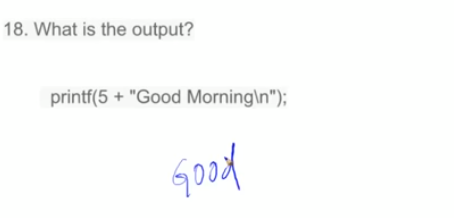
**int STATIC =90;**

**int Integer =20;**

**float String=9;**

**int Character=90;**

**double Double=88;**

****

**18. it is not array so direct start location from 1 and take upto 5th location, start with 1 ,so print good and blank space.**