Session Goals

- User should understand strings and immutability of strings in Java.
- User should understand the need stringBuilder class in Java and how it helps in making string mutable.
- User should be able to convert StringBuilders to strings and vice versa.
- User should be able to convert Integers into String and vice versa.

Java-111- String and StringBuilder in Java

Session 8

Session Agenda

- Strings in Java
  - Immutability of String
  - String library methods
  - Converting to and from Strings
- StringBuilder

Creating Strings

In Java, **String** is basically an object that represents a sequence of char values. The **java.lang.String** class is used to create a String object.

There are 3 ways to create a String object

1. Using a **String Literal**
2. Converting from a **char array**
3. Using the **new** keyword

```
public class StringExample{
  public static void main(String args[]){
    String s1="java";                     //creating string by Java string literal
    char ch[]={'s','t','r','i','n','g','s'};
    String s2=new String(ch);             //converting char array to string
    String s3=new String("example");//creating Java string by new keyword
    System.out.println(s1);
    System.out.println(s2);
    System.out.println(s3);
  }
}
```

Immutability of String

String is a Class. When you create a new String, you are creating a new String Class Object.

String values are **immutable**, which means that they cannot be altered once created.

```
String myStr = "Bob";
myStr = "Uncle Bob";
```

How does it work without error?

```
String myStr = "Bob";   // Note that “Bob” is the object. “myStr” is the reference to the Object.
myStr = "Uncle Bob";   // New Object “Uncle Bob” is created here and the “myStr” now points to the new Object
```

Summary => Any new assignment or update creates a new String Object.

Curious Cats

```
void foo(String errorText){
    errorText += "error";
}

int main(){
    String error="Overflow";
    foo(error);
    System.out.println(error);
}
```

What’s the expected output here?

```
String foo(String errorText){
    return errorText + "error";
}

int main(){
    String error="Overflow";
    error = foo(error);
    System.out.println(error);
}
```

What’s the expected output here?

Guess the output

```java
public class StringExample
{
  public static void main(String[] args)
  {
    String s1 = "Bob";
    String s2 =  s1;
    System.out.println((s1 == s2));
    s2 = "Uncle Bob";
    System.out.println((s1 == s2));

    System.out.println(s1);
    System.out.println(s2);
  }
}
```

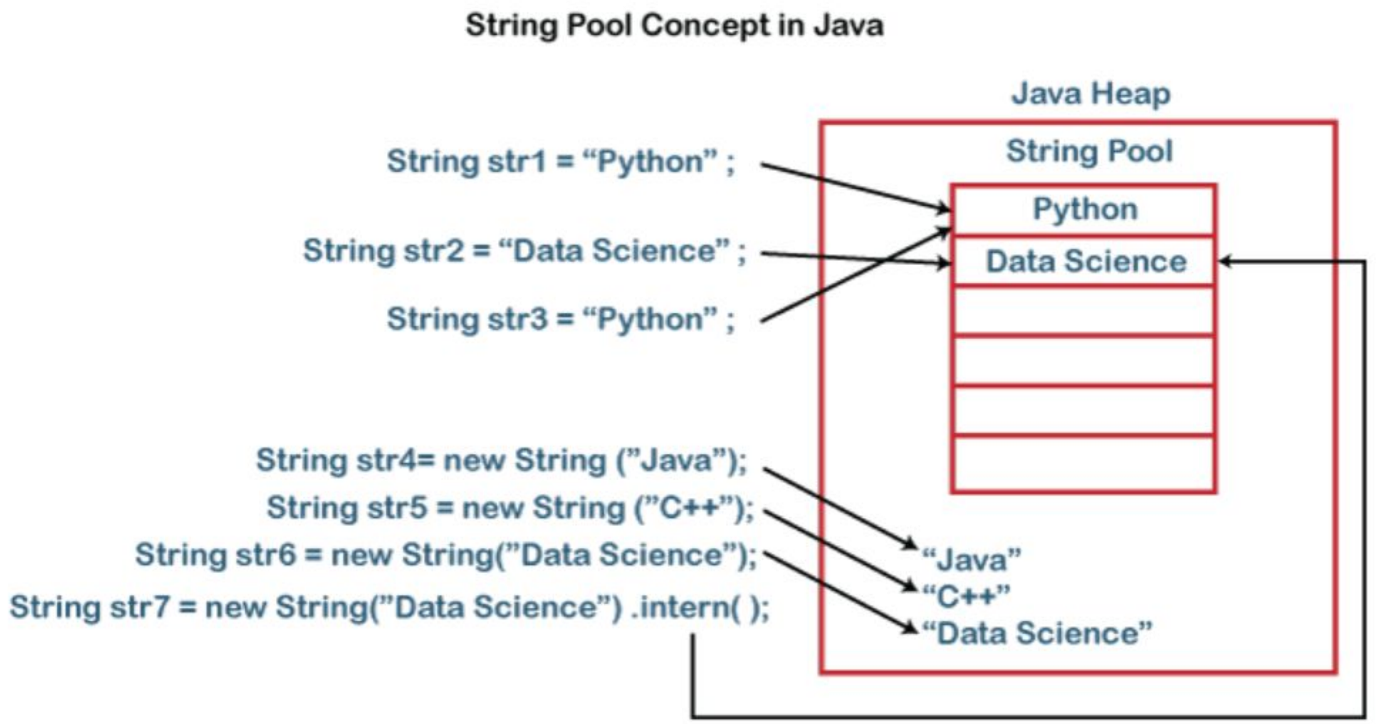
```java
public class StringExample
{
  public static void main(String[] args)
  {
    String s1 = "Bob";
    String s2 =  s1;
    System.out.println((s1 == s2)); // true
    s2 = "Uncle Bob";
    System.out.println((s1 == s2)); // false

    System.out.println(s1); // Bob
    System.out.println(s2); // Uncle Bob
  }
}
```

Activity - == vs .equals()

```
public class StringPoolExample
{
    public static void main(String[] args)
    {
        String s1 = "Java";
        String s2 = "Java";
        String s3 = new String("Java");
        System.out.println((s1 == s2)+", String are equal."); // true
        System.out.println((s1 == s3)+", String are not equal."); // false
    }
}
```

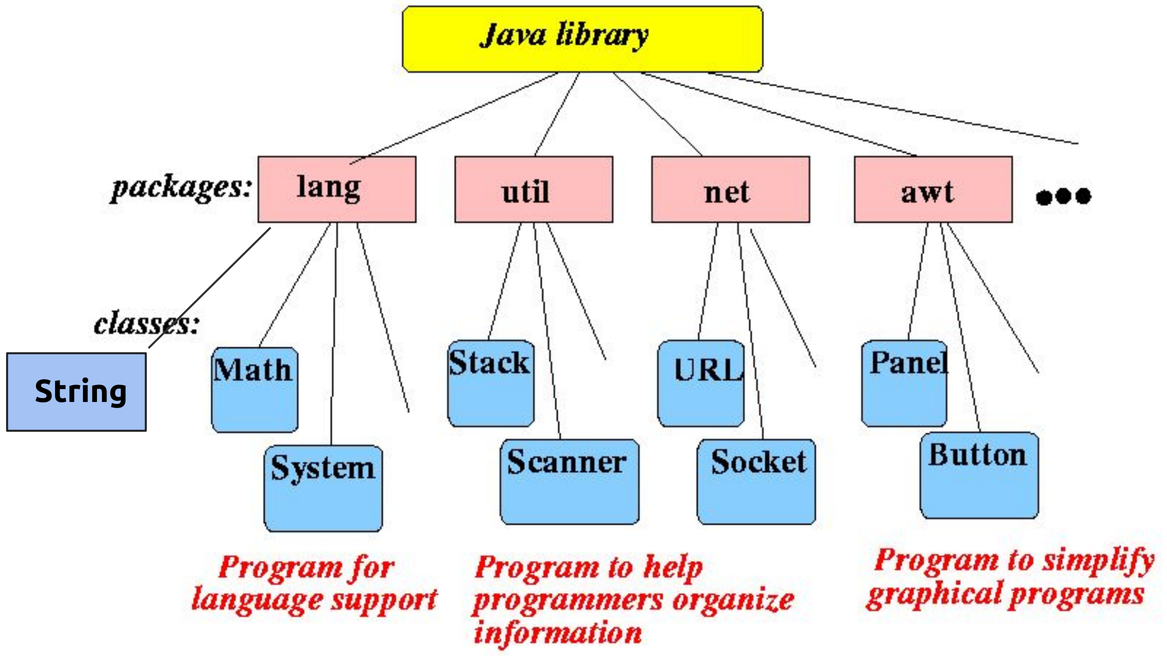
String Pool in Java


String Pool Concept in Java
Java Heap
String Pool
String str1 = "Python" ;
String str2 = "Data Science" ;
String str3 = "Python" ;
Python
Data Science
String str4= new String ("Java");
String str5 = new String ("C++");
String str6 = new String("Data Science");
String str7 = new String("Data Science") .intern( );
"Java"
"C++"
"Data Science"

String Pool in Java | Debrief

- What is a **String Pool**?
  - String pool is a **storage area on the Heap** where **string literals are stored**. Specific to each program.
- Why is it needed?
  - String objects take **space** to be stored in memory.
  - JVM performs some **optimization** to reduce this memory usage. To decrease the number of String objects created in the JVM, the **String class keeps a Pool of Strings**.
  - When we create a string literal, the JVM **first checks if that literal is present** in the String Pool. If the literal is **already present, it returns a reference** to the pooled instance. If the literal **is not present in the pool, a new String object gets created** in the String pool.

Concept - String class from Java Library



Java library
packages:
lang
util
net
awt
•••
classes:
String
Math
System
Stack
Scanner
URL
Socket
Panel
Button
Program for language support
Program to help programmers organize information
Program to simplify graphical programs

Concept #3 - String class and its methods

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Learn to read Standard Documentation. Some key methods:

- charAt()
- indexOf()
- endsWith()
- startsWith()
- contains()
- length()
- replace()
- substring()
- toLowerCase()
- toUpperCase()
- valueOf()
- concat()
- equals() - we've already seen this
- trim()

Recap - 6 Step Strategy

1. ***Understand the problem*** *(ask questions and get clarity)*
2. ***Design test data/test cases*** *(input and expected output)*
3. ***Derive the solution - solve the problem*** *(write pseudo code)*
4. ***Test the solution*** *(against the test data/case - dry run)*
5. ***Write the program/code*** *(using Java here)*
6. ***Test the code*** *(syntax errors, run time errors, logical errors)*

Activity: Check if the next animal is a mouse

Link

What will be your approach to the problem? (Step 3)

Quickly put your answers in the chat!

5 minute break

Converting to and from Strings

- .toString()
  - The toString() method returns the string representation of the object.
  - Most inbuilt wrapper classes support this method. E.g. Integer.toString() *(We will visit wrapper classes in the next session)*
- .valueOf()
  - The java String valueOf() method converts different types of values into string.
  - By the help of string valueOf() method, you can convert int, long, boolean, character, float, double, object or char array to string.
  - Similarly, there also exists Integer.valueOf() method etc.

Activity: Convert a Number to string

Link

What will be your approach to the problem? (Step 3)

Quickly put your answers in the chat!

String Builder

- The **StringBuilder** in Java represents a **mutable sequence** of characters.
- Since the **String** Class in Java creates an immutable sequence of characters, the StringBuilder class provides an **alternative** to String Class, as it creates a mutable sequence of characters.
- .append(), .reverse, .insert(), .replace(), .delete() etc.

```
class StringBuilderExample{
  public static void main(String args[]){
    StringBuilder sb=new StringBuilder("Hello ");
    sb.append("Java");//now original string is changed
    System.out.println(sb);//prints Hello Java
  }
}
```

Activity: Add Spaces between Words

Link

What will be your approach to the problem? (Step 3)

Quickly put your answers in the chat!

Activity: Reverse a string

Link

What will be your approach to the problem? (Step 3)

Quickly put your answers in the chat!

String Templates

- Simple way to format String data
  - String message = "Hello" + " World";
- A template is a *String* that contains some static text and one or more format specifiers, which indicate which argument is to be placed at the particular position.
- String.format()
  - String message = String.format("Hello! My name is %s, I'm %s.", name, age);
  - Different format specifiers (%s, %d, %f, %c etc.) can be used

Further Reading - https://www.baeldung.com/java-string-formatter

Further Reading

- Java String is Immutable
- String Pool in Java
- String Methods
- StringBuilder
- StringBuffer
- String Template
- Command line arguments

Keep
Learning,
Keep
Coding.