OOP using Java

Trainer: Mr. Rohan Paramane

Agenda

- Multi Dimension Array
- Ragged Array
- Variable Arity/Argument Method
- String
- Enum

Multi Dimensional Array

- Array of elements where each element is array of same column size is called as multi dimensional array.

Reference declaration:

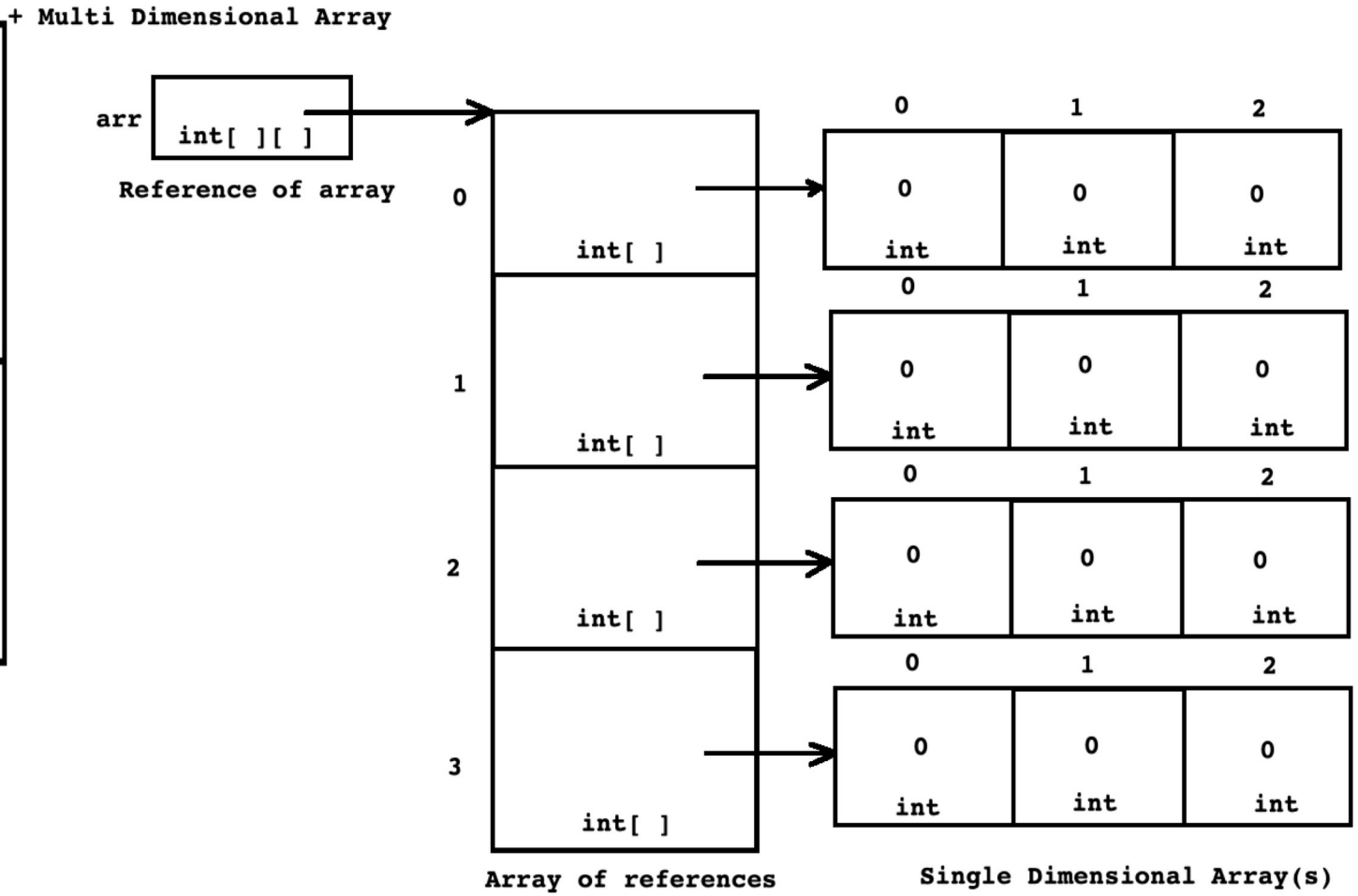
```
int arr[ ][ ];  //OK
int [ ]arr[ ]   //OK
int[ ][ ] arr;  //OK
```

Array Creation:

```
int[][] arr = new int[ 2 ][ 3 ];
```

Initialization

```
int[][] arr = new int[ ][ ]{{10,20,30},{40,50,60}}; //OK
int[][] arr = { {10,20,30}, {40,50,60} };    //OK
```



Ragged Array

- A multidimensional array where column size of every array is different.

Reference declaration

```
int arr[][];
int []arr[];
int[][] arr;
```

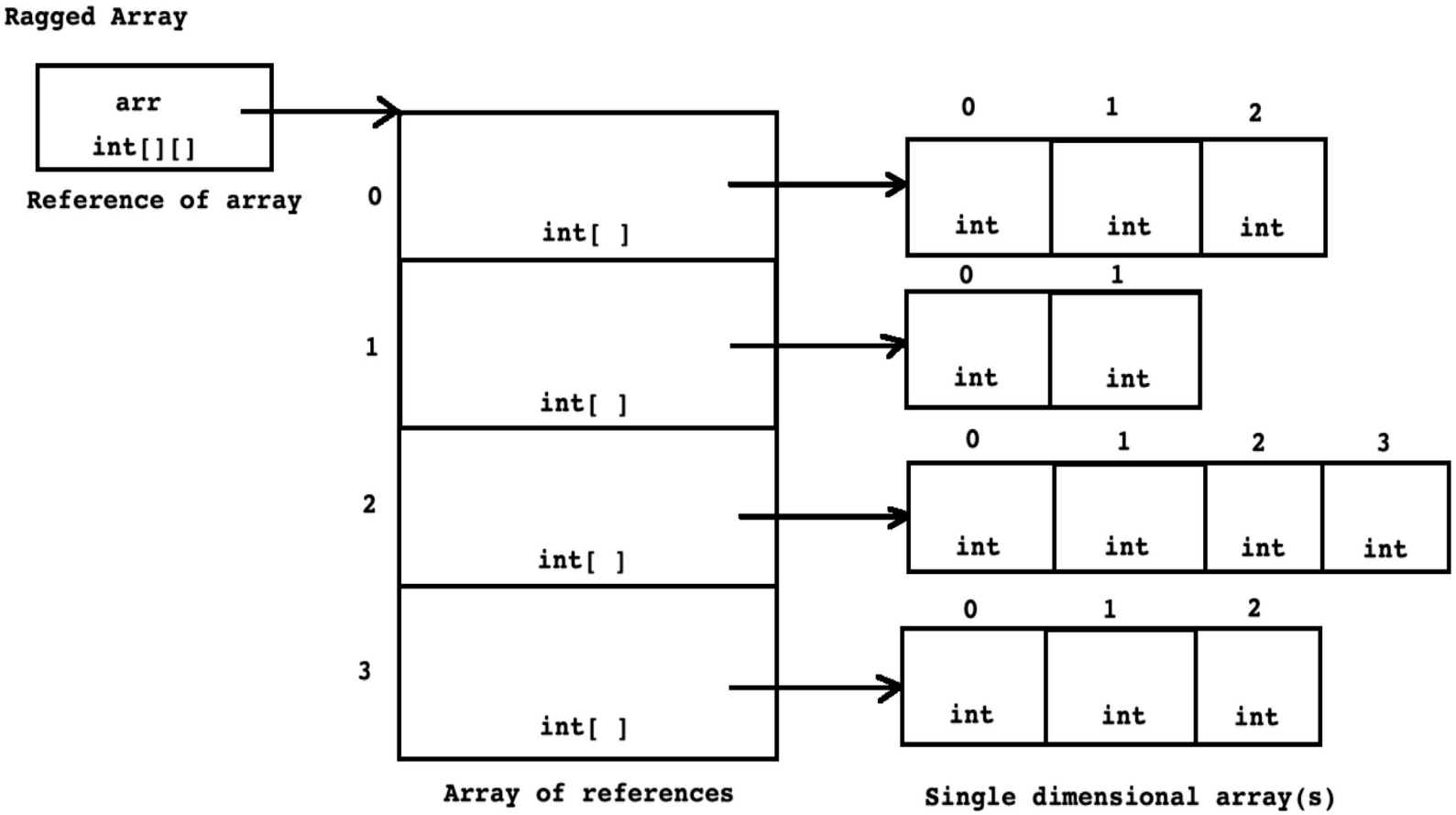
Array creation

```
int[][] arr = new int[3][];
arr[ 0 ] = new int[ 2 ];
arr[ 1 ] = new int[ 3 ];
arr[ 2 ] = new int[ 5 ];
```

Array Initialization

```
int[][] arr = new int[3][];
arr[ 0 ] = new int[  ]{ 10, 20 };
arr[ 1 ] = new int[  ]{ 10, 20, 30 };
arr[ 2 ] = new int[  ]{ 10, 20, 30, 40, 50 };
```

```
int[][] arr = { { 1, 2 }, { 1, 2, 3 }, {1,2,3,4,5}};
```



Variable Arity/Argument Method

```
private static sum( int... arguments ){
    int result = 0;
    for( int element : arguments )
        result = result + element;
    return result;
}
public static void main(String[] args) {
    int result = 0;
    result = Program.sum( );     //OK
    result = Program.sum( 10, 20, 30 );     //OK
    result = Program.sum( 10, 20, 30, 40, 50  );     //OK
    result = Program.sum( 10, 20, 30, 40, 50, 60, 70, 80, 90, 100  );     //OK
}
```

String Introduction

- String is not a built-in or primitive type. It is a class, hence considered as non primitive/reference type.
- We can create instance of String with and without new operator.
  - String str = “Sunbeam”
  - String str = new String(“Rohan”);
- Two ways to create a String in Java which are String Literal and String Object.
- The main difference between String Literal and String Object is –
  - String Literal is String created using double quotes
  - String Object is a String created using the new() operator.
- Strings, which are widely used in Java programming, are a sequence of characters. In the Java programming language, strings are objects.
- We can use following classes to manipulate string
  - java.lang.String
  - java.lang.StringBuffer
  - java.lang.StringBuilder
  - java.util.StringTokenizer

Literal String

- **String s1 = “Hello World”;**
- Here, the s1 is referring to “Hello World” in the String pool.
- If there is another statement as follows.
- **String s2 = “Hello World”;**
- As “Hello World” already exists in the String pool, the s2 reference variable will also point to the already existing “Hello World” in the String pool. In other words, now both s1 and s2 refer to the same “Hello World” in the String pool.
- Therefore, if the programmer writes a statement as follows, it will display true.
- **System.out.println(s1==s2);**
- String Pool in Java is a special storage space in Java heap memory. It is also known as String Constant Pool or String Intern Pool.
- Whenever a new string is created, JVM first checks the string pool. If it encounters the same string, then instead of creating a new string, it returns the same instance of the found string to the variable.
- The String.intern() method puts the string in the String pool or refers to another String object from the string pool having the same value.

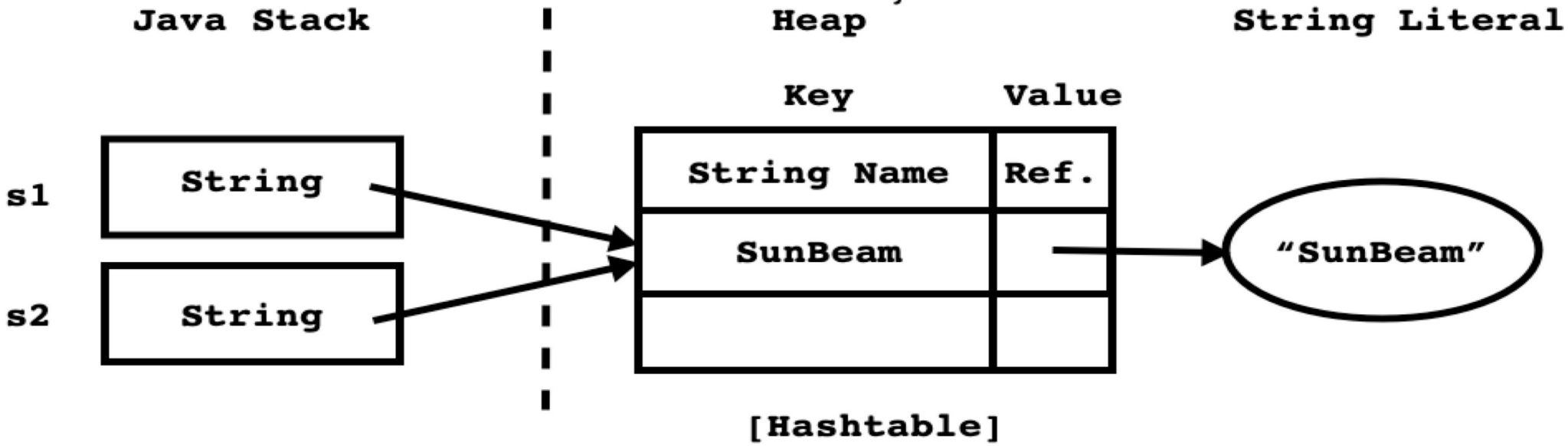
Non Literal String

- **String s1 = new (“Hello World”);**
- **String s2 = new (“Hello World”);**
- Unlike with String literals, in this case, there are two separate objects.
- In other words, s1 refers to one “Hello World” while s2 refers to another “Hello World”.
- Here, the s1 and s2 are reference variables that refer to separate String objects.
- Therefore, if the programmer writes a statement as follows, it will display false.
- **System.out.println(s1==s2);**

String Examples

```java
public class Program {
   public static void main(String[] args) {
      String s1 = "SunBeam";
      String s2 = "SunBeam";
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Equal
   }
}
```

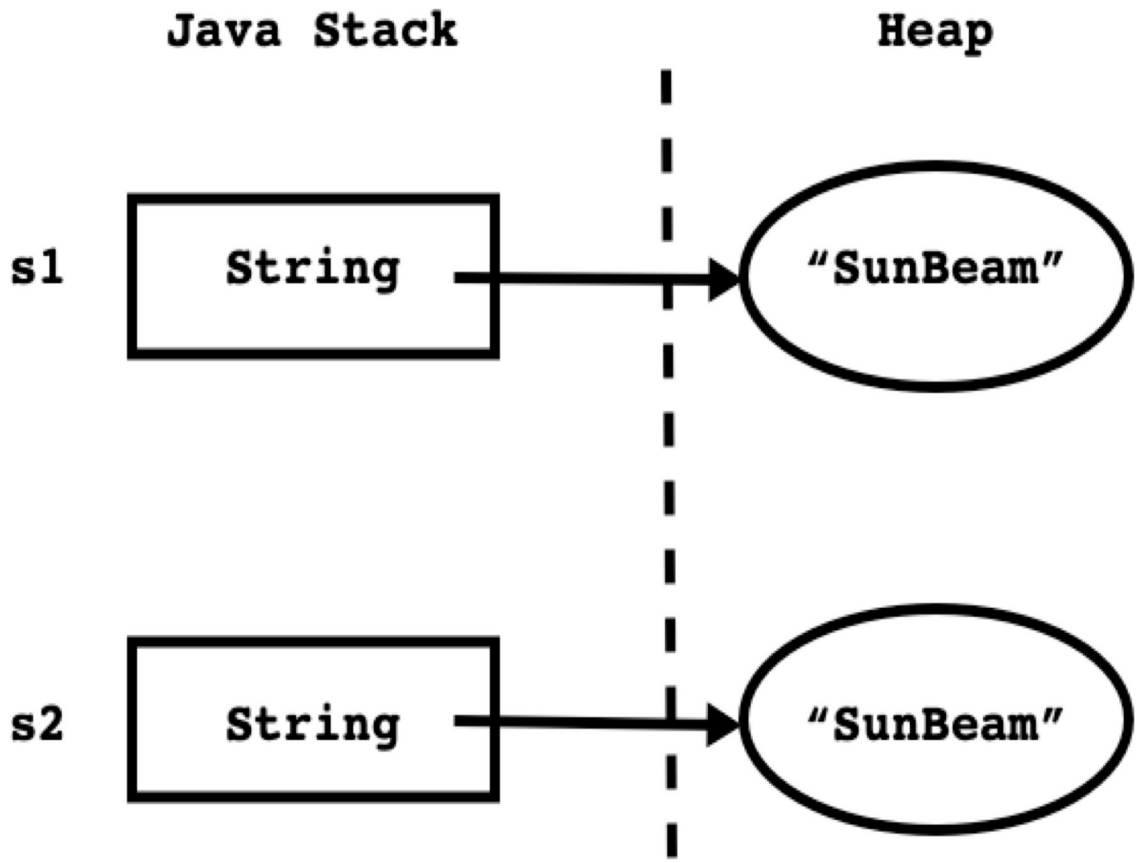
```java
public class Program {
   public static void main(String[] args) {
      String s1 = "SunBeam";
      String s2 = "SunBeam";
      if( s1.equals(s2) )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Equal
   }
}
```



String Examples

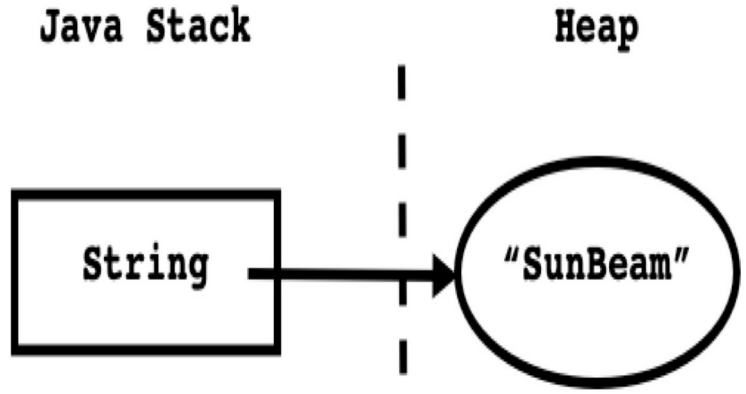
```
public class Program {
   public static void main(String[] args) {
      String s1 = new String("SunBeam");
      String s2 = new String("SunBeam");
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      String s1 = new String("SunBeam");
      String s2 = new String("SunBeam");
      if( s1.equals(s2) )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Equal
   }
}
```

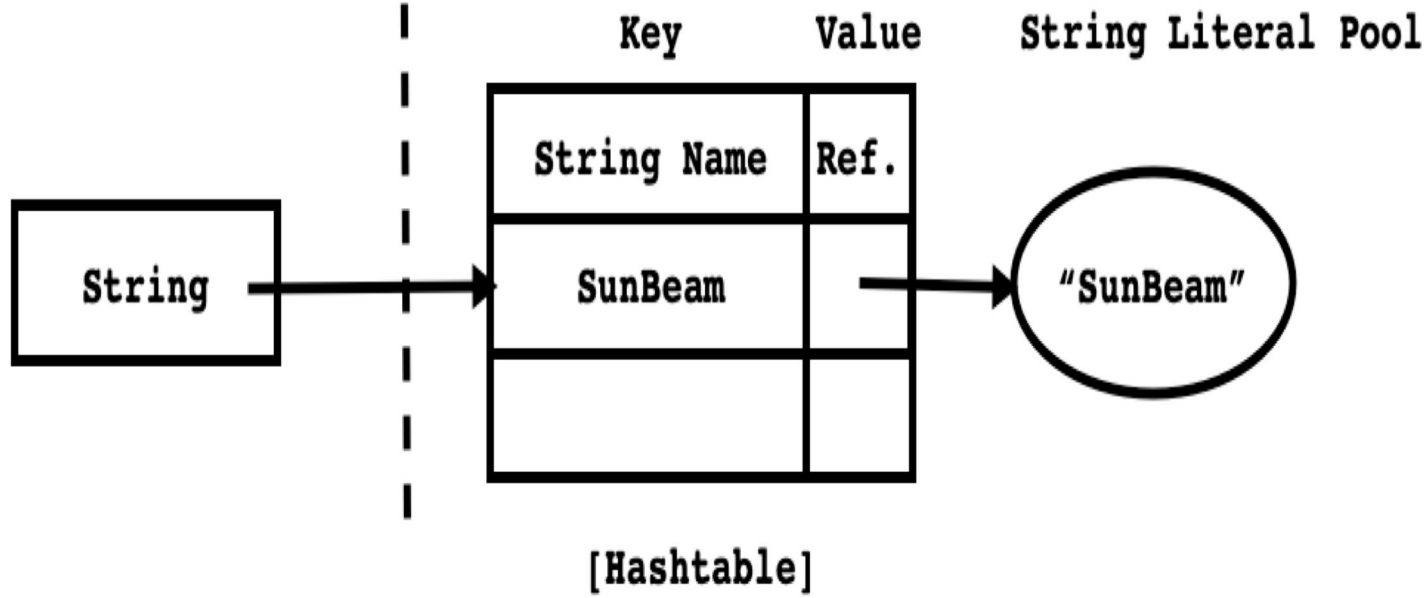


String Examples

```java
public class Program {
   public static void main(String[] args) {
      String s1 = new String("SunBeam");
      String s2 = "SunBeam";
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```




```java
public class Program {
   public static void main(String[] args) {
      String s1 = new String("SunBeam");
      String s2 = "SunBeam";
      if( s1.equals(s2) )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Equal
   }
}
```



String Examples

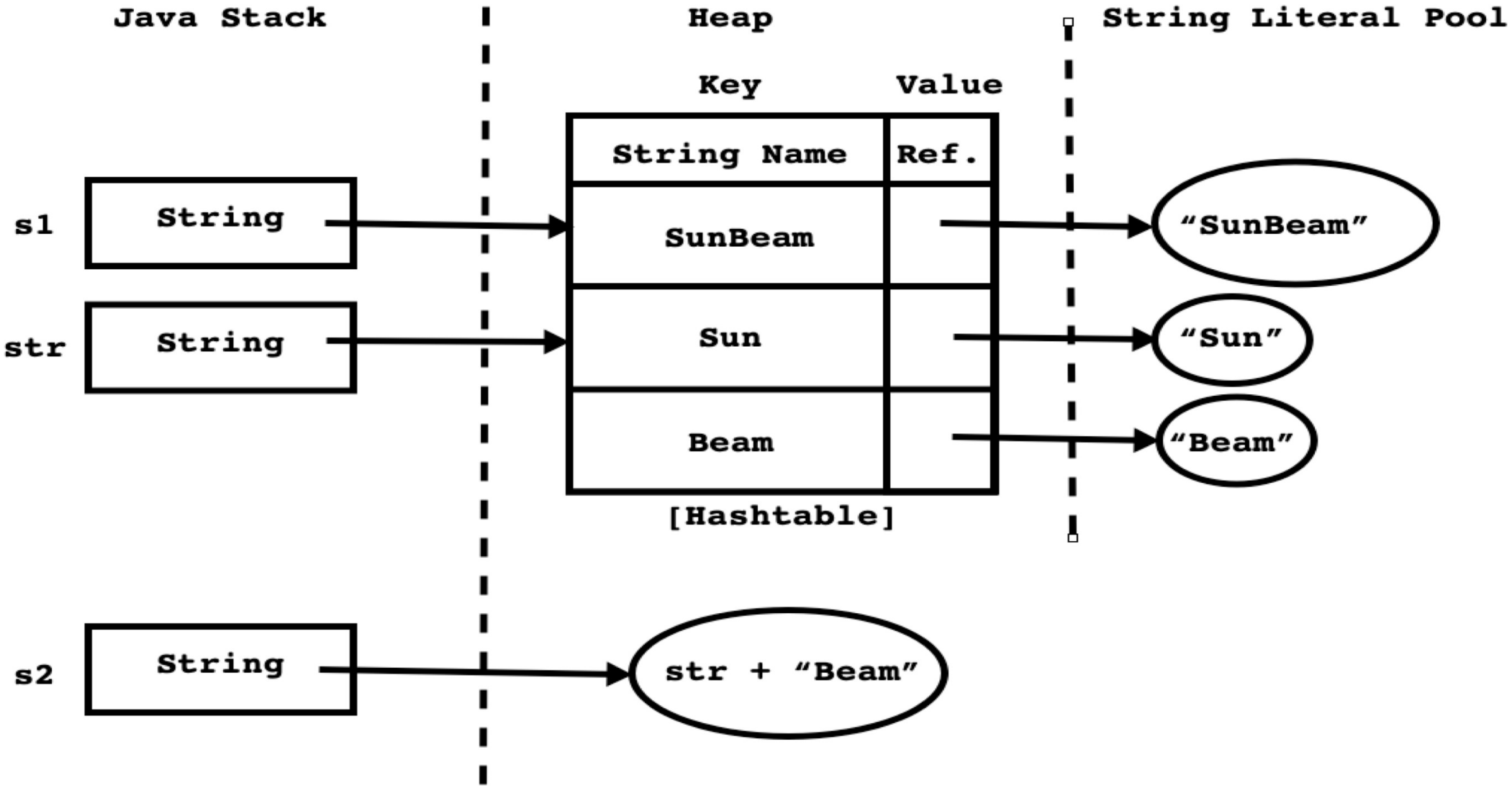
- Constant expression gets evaluated at compile time.
- “int result = 2 + 3;” becomes “int result = 5;” at compile time
- ”String s2 = “Sun”+”Beam”;” becomes “String s2=“SunBeam”;” at compile time.

```
public class Program {
   public static void main(String[] args) {
      String s1 = "SunBeam";
      String s2 = "Sun"+"Beam";
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Equal
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      String s1 = "SunBeam";
      String str = "Sun";
      String s2 = str + "Beam";
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      String s1 = "SunBeam";
      String str = "Sun";
      String s2 = ( str + "Beam" ).intern();
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Equal
   }
}
```

String Examples



String Examples

```java
package p1;
public class A {
    public static final String str = "Hello";
}
```

```java
package test;
class B {
    public static final String str = "Hello";
}
```

```java
package test;
public class Program {
    public static final String str = "Hello";
    public static void main(String[] args) {
        String str = "Hello";
    }
}
```

```java
public class Program {
    public static final String str = "Hello";
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println(A.str == B.str);          //true
        System.out.println(A.str == Program.str);    //true
        System.out.println(A.str == str);           //true
        System.out.println(B.str == Program.str);    //true
        System.out.println(B.str == str);           //true
        System.out.println(Program.str == str);      //true
    }
}
```

StringBuffer versus StringBuilder

- StringBuffer and StringBuilder are final classes.
- It is declared in java.lang package.
- It is used create to mutable string instance.
- equals() and hashCode() method is not overridden inside it.
- We can create instances of these classes using new operator only.
- Instances get space on Heap.
- StringBuffer implementation is thread safe whereas StringBuilder is not.
- StringBuffer is introduced in JDK1.0and StringBuilder is introduced in JDK 1.5.

StringBuffer Example

```
public class Program {
   public static void main(String[] args) {
      StringBuffer s1 = new StringBuffer("SunBeam");
      StringBuffer s2 = new StringBuffer("SunBeam");
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      String s1 = new String("SunBeam");
      StringBuffer s2 = new StringBuffer("SunBeam");
      if( s1 == s2 )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Compiler Error
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      StringBuffer s1 = new StringBuffer("SunBeam");
      StringBuffer s2 = new StringBuffer("SunBeam");
      if( s1.equals(s2) )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      String s1 = new String("SunBeam");
      StringBuffer s2 = new StringBuffer("SunBeam");
      if( s1.equals(s2) )
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

StringBuilder Example

```
public class Program {
   public static void main(String[] args) {
      StringBuilder s1 = new StringBuilder("SunBeam");
      StringBuilder s2 = new StringBuilder("SunBeam");
      if( s1 ==  s2)
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

```
public class Program {
   public static void main(String[] args) {
      StringBuilder s1 = new StringBuilder("SunBeam");
      StringBuilder s2 = new StringBuilder("SunBeam");
      if( s1.equals(s2))
         System.out.println("Equal");
      else
         System.out.println("Not Equal");
      //Output : Not Equal
   }
}
```

StringTokenizer

```
public class Program {
   public static void main(String[] args) {
      String str = "SunBeam Infotech Pune";
      StringTokenizer stk = new StringTokenizer(str);
      String token = null;
      while( stk.hasMoreTokens()) {
         token = stk.nextToken();
         System.out.println(token);
      }
   }
}
```

```
Output
---------
SunBeam
Infotech
Pune
```

```
public class Program {
   public static void main(String[] args) {
      String str = "www.sunbeaminfo.com";
      String delim = ".";
      StringTokenizer stk = new StringTokenizer(str, delim);
      String token = null;
      while( stk.hasMoreTokens()) {
         token = stk.nextToken();
         System.out.println(token);
      }
   }
```

```
Output
---------------
www
sunbeaminfo
com
```

String Tokenizer

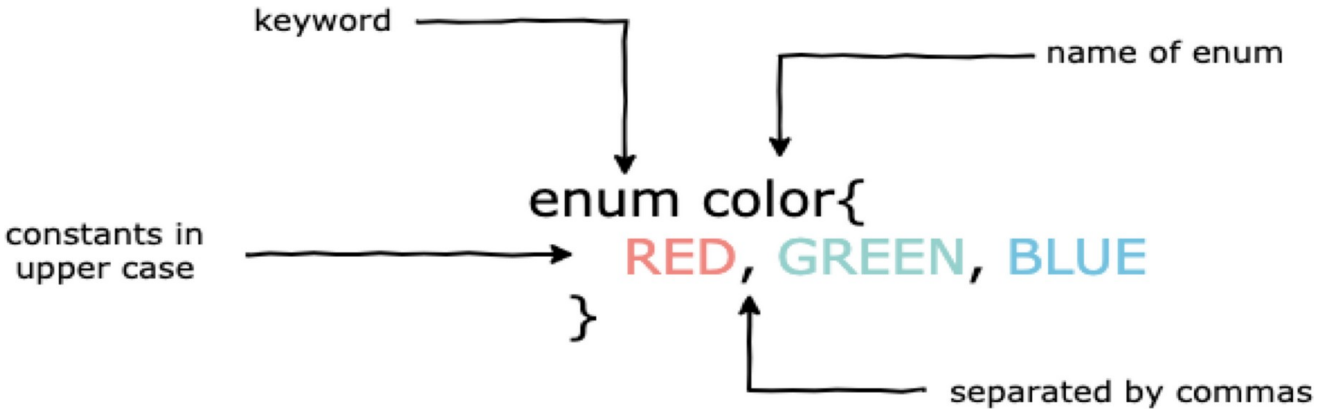
```java
import java.util.Scanner;
import java.util.StringTokenizer;


public class Day6_5
{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    {
        String str = "https://admission.sunbeaminfo.com/aspx/RegistrationForm.aspx?BatchID=J8BwSw7MbJHgHVtHZgIUlA==";

        String delim = "/:-=.//#";
        StringTokenizer stk = new StringTokenizer(str, delim, true);

        String token  = null;
        while( stk.hasMoreTokens()) {
            token = stk.nextToken();
            System.out.println(token);
    }
}
}
```

OUTPUT

```
https
:
/
/
admission
.
sunbeaminfo
.
com
/
aspx
/
RegistrationForm
.
aspx?BatchID
=
J8BwSw7MbJHgHVtHZ
gIUlA
=
=
```

Enum

- According ANSI C standard, if we want to assign name to the integer constant then we should use enum.
- Enum helps developer to improve readability of source code.
- An enum is a class that represents a group of constants
- Enum keyword is used to create an enum. The constants declared inside are separated by a comma and should be in upper case.
- enum is used for values that are not going to change e.g. names of days, colors in a rainbow, number of cards in a deck etc.
- enum is commonly used in switch statements.



Enum

- Similar to a class, an enum can have objects and methods. The only difference is that enum constants are public, static and final by default. Since it is final, we can't extend enums
- It cannot extend other classes since it already extends the java.lang.Enum class.
- It can implement interfaces.
- The enum objects cannot be created explicitly and hence the enum constructor cannot be invoked directly.
- It can only contain concrete methods and no abstract methods.

Java Source Code

```
enum Color{
    RED, GREEN, BLUE
}
class Program{
    public static void main(String[] args) {
        Color color = Color.GREEN;
    }
}
```

Compiled Code

```
final class Color extends Enum<Color> {
    public static final Color RED;

    public static final Color GREEN;

    public static final Color BLUE;

    public static Color[] values();

    public static Color valueOf(String name);
}
```

Thank you!

Rohan Paramane

rohan.paramane@sunbeaminfo.com