PSOOP(JAVA) – LECTURE 05
STRINGS AND STATIC IN JAVA

-Compiled by Nikahat Mulla

Agenda

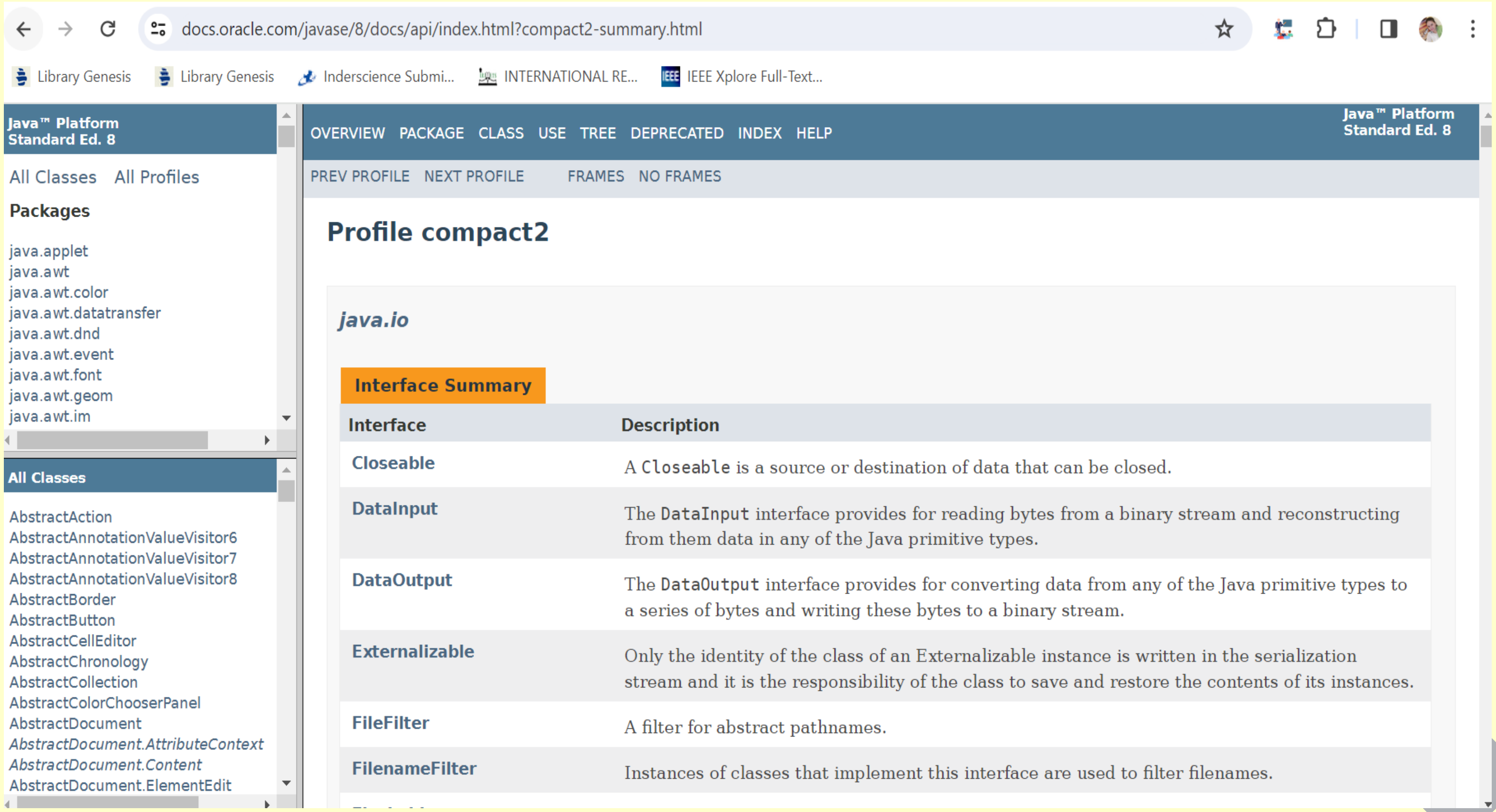
- Strings in Java
- static keyword

Using *String* Class

- Present in *java.lang* package
- An object of the String class represents a fixed length, immutable sequence of characters
- Has equals( ) method that should be used to compare the actual string values
- A lot of other string manipulation methods are available
- JavaDocs can be referred for a detailed list of methods

Referring Java Documentation

- Java provides a rich set of library classes
- Java API Documentation provides detailed help on all classes
- Browse Java API Documentation



Profile compact2

java.io

Interface Summary

Interface	Description
Closeable	A `Closeable` is a source or destination of data that can be closed.
DataInput	The `DataInput` interface provides for reading bytes from a binary stream and reconstructing from them data in any of the Java primitive types.
DataOutput	The `DataOutput` interface provides for converting data from any of the Java primitive types to a series of bytes and writing these bytes to a binary stream.
Externalizable	Only the identity of the class of an Externalizable instance is written in the serialization stream and it is the responsibility of the class to save and restore the contents of its instances.
FileFilter	A filter for abstract pathnames.
FilenameFilter	Instances of classes that implement this interface are used to filter filenames.

Constructors of String class

- First Constructor takes an array of char type and converts it into a String object
- Second Constructor reads a byte[] array and converts into a String object
- Third Constructor converts a String literal into a String object

String (char[])

- char ch={'H', 'E', 'L', 'L', 'O'};
- String s1=new String(ch);

String (byte[])

- byte b[]={65,66,67,68};
- String s2=new String(b);

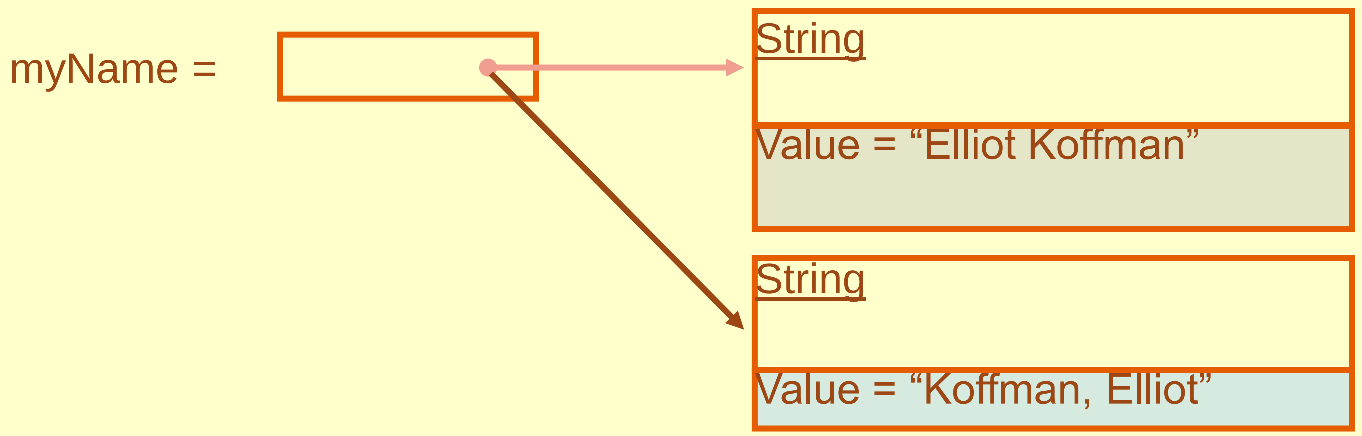
String(String)

- String s3=new String("Java");
- "Java" is a string literal->saved in the literal pool
- s3 is an object in which the string literal Java is copied
- Above loc creates 2 objects, one for the literal and one for s3

Using *String* Class (Contd…)

- Defines a data type used to store a sequence of characters
- Strings are objects
- String objects can’t be modified:
  - If attempted to do so, Java creates a new object having the modified character sequence. For e.g.,

```
String myName = "Elliot Koffman";
myName = "Koffman, Elliot";
```



Methods of String class

- int length()
- String toLowerCase()
- String toUpperCase()
- String trim()
- String substring(int begin)
- String substring(int beg,int end)
- String replace(char old,char new)
- boolean startsWith(String s)
- boolean endsWith(String s)
- char charAt(int idx)
- int indexOf(String s)
- int lastIndexOf(String s)
- boolean equals(String s)
- boolean equalsIgnoreCase(String s)
- int compareTo(string)
- String valueOf(int i)

COMMON STRING OPERATIONS

- String concatenation

```
String u = “Hello”;
String t = “ World”;
String s = u + t; // s refers to “Hello World”
int i = s.length();  // returns 11
u.equals(t)          // comparison, returns false
u.compareTo(t)      // returns negative number
s.charAt(1)         // returns ‘e’, index runs
                 //from 0 to length-1
String x = u.toUpperCase();   //returns “HELLO”
```

USING *STATIC*

- *static* keyword can be used in three scenarios:
  - For class variables
  - For methods
  - For a block of code

note

Using *static* (Contd…)

- *static variable*
  - Belongs to a class
  - A single copy to be shared by all instances of the class
  - Creation of instance not necessary for using static variables
  - Accessed using *<class-name>.<variable-name>* unlike instance variables which are accessed as *<object-name>.<variable-name>*
- *static method*
  - It is a class method
  - Accessed using *class name.method name*
  - Creation of instance not necessary for using static methods
  - A static method can access only other static data & methods, and not non-static members

note

USING *STATIC* (CONTD…)

```
class Student {
   private int rollNo;
   private static int studCount;
   public Student(){
         studCount++;
   }
public void setRollNo (int r){
        rollNo = r;
   }
   public int getRollNo (int r){
        return rollNo;
   }
   public static void main(String args[]){
        System.out.println("RollNo of the Student is;" +
   rollNo);
   }
}
```

The static studCount variable is initialized to 0, ONLY when the class is first loaded, NOT each time a new instance is made

Each time the constructor is invoked, i.e. an object gets created, the static variable studCount will be incremented thus keeping a count of the total no of Student objects created

Which Student? Whose rollNo? A static method cannot access anything non-static

Static Block

- Java supports a special block, called a static block (also called static clause) that can be used for static initialization of a class.
- This code inside the static block is executed only once: the first time the class is loaded into memory
- static block executes automatically when the class is loaded in memory.
- Refer: TestStaticBlock.java

Thank You