DAY-49

07 September 2023 15:49

```
package string;

public class StringMethods {

        public static void main(String[] args)
        {
                StringBuffer s1 = new StringBuffer("java");
                StringBuffer s6 = new StringBuffer("java");
                s1.append(" class");

                System.out.println("String Buffer");
                System.out.println(s1.hashCode());
                System.out.println(s6.hashCode());
                System.out.println(s1.equals(s6));

                StringBuilder s2 = new StringBuilder("manual");
                StringBuilder s5 = new StringBuilder("manual");
                s2.append(" Testing");

                System.out.println("StringBuilder");
                System.out.println(s2.hashCode());
                System.out.println(s5.hashCode());
                System.out.println(s2.equals(s5));

                String s3 = new String("html");
                String s4 = new String("html");
                s3.concat(" class");

                System.out.println("String class");
                System.out.println(s3.hashCode());
                System.out.println(s4.hashCode());
                System.out.println(s3.equals(s4));


        }

}
```

- toString() OF OBEJCT CLASS IS OVERRIDDEN IN STRING, STRINGBUILDER AND STRING BUFFER BUT ,
- hashCode() and equals() OF OBJECT CLASS ARE OVERRIDDEN IN STRING BUT NOT OVERRIDDEN IN StringBuffer AND StringBuilder.

*****WHAT IS DIFFERENCE BETWEEN STRING, STRING BUFFER AND STRING BUILDER ?

STRING	STRING BUFFER	STRING BUILDER
IT IS IMMUTABLE CLASS	IT IS MUTABLE CLASS	IT IS MUTABLE CLASS
toString(), hashCode() and equals()	toString() IS OVERRIDDEN BUT	toString() IS OVERRIDDEN BUT

OF OBJECT CLASS ARE OVERRIDDEN	hashCode() and equals() ARE NOT OVERRIDDEN.	hashCode() and equals() ARE NOT OVERRIDDEN.
OBJECT CAN BE CREATED WITH OR WITHOUT THE NEW KEYWORL.	OBJECT HAS TOBE CREATED WITH USING THE NEW KEYWORD	OBJECT HAS TO BE CREATED WITH USING THE NEW KEYWORD.

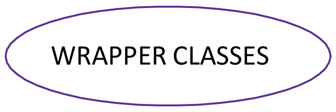
*****WHAT IS DIFFENCE BETWEEN STRING BUFFER AND STRING BUILDER

STRING BUFFER	STRING BUILDER
StringBuffer IS LESS EFFICEINT THAN StringBuilder.	StringBuilder IS MORE EFFICIENT THAN StringBuffer.
StringBuffer IS SYNCHRONIZED	StringBuilder IS NOT SYNCHRONIZED
IT IS THREAD SAFE , i.e, TWO THREADS CAN NOT CALL THE METHODS OF StringBuffer SIMOUTANEOUWRSLY	IT IS NOT THREAD SAFE i.e, TWO THREADS CAN CALL THE METHODS OF StringBuilder SIMULTANEOUSLY.

WRAPPER CLASSES

WHAT ARE WRAPPER CLASSES ?

- THESE ARE USED TO CONVERT PRIMITIVE TYPE TO NON PRIMITIVE OR NON PRIMITIVE TYPE TO PRIMITIVE TYPE.
- EACH PRIMITIVE TYPE HAS ITS CORRESPONDING WRAPPER CLASSES.
- ALL WRAPPER CLASSES ARE AVAILABLE IN JAVA.LANG PACKAGE
- ALL WRAPPER CLASSES ARE FINAL.
- ALL WRAPPER CLASSES OVERRIDE toString(), hashCode() AND equals(Object obj) OF OBEJCT CLASS.

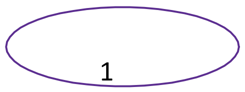


JAVA.LANG PACKAGE
ALL WRAPPER CLASSES ARE FINAL

WRAPPER CLASSES ARE USED TO CONVERT PRIMITIVE DATATYPE TO NON PRIMITIVE DATATYPE AND VICE VERSA

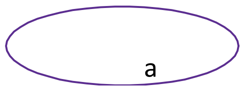
PRIMITIVE DATATYPE	WRAPPER CLASS
byte	Byte.java
short	Short.java
int	Integer.java
long	Long.java
double	Double.java
float	Float.java
char	Character.java
boolean	Boolean.java

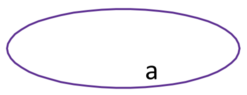
1 primitive datatype

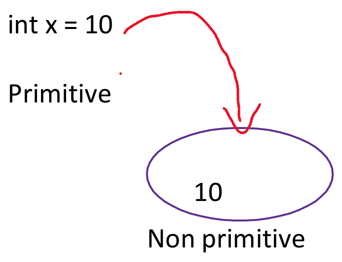


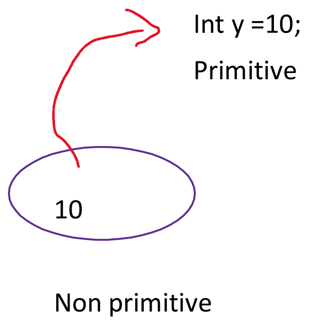
Non primitive

char c = 'a'









- BOXING : CONVERTING PRIMITIVE TYPE OF DATA TO NON PRIMITIVE TYPE OF DATA
- UNBOXING : CONERTING NON PRIMITIVE TYPE OF DATA TO PRIMITIVE TYPE OF DATA

BOXING AND UNBOXING CAN BE DONE IMPLICITILY BY JVM AND ALSO EXPLICITILY BY USER.

```
package string;

public class Box {

	public static void main(String[] args)
	{
		int x = 1;
		  Integer y = x;//implicit boxing
		  System.out.println(y);
		  System.out.println(y.hashCode());

		  double z = 2.1;
		  Double a = Double.valueOf(z); //explicit boxing
		  System.out.println(a);
		  System.out.println(a.hashCode());

		  int x1 = y; //implicit unboxing
		  System.out.println(x1);

		  double a1 = a.doubleValue();//explicit unboxing
		  System.out.println(a1);

	}

}
```

<u>**ADVANTAGES OF WRAPPER CLASSES**</u>

- SINCE COLLECTION WILL ONLY STORE OBJECTS/CLASS TYPE VALUES, IF WE WANT TO STORE PRIMITIVE VALUES IN THE COLLECTION THEN WE HAVE TO CONVERT PRIMITIVE TO NON PRIMITIVE.

- THIS CONVERSION FROM PRIMITIVE TO NON PRIMITIVE OR VICE VERSA CAN HAPPEN ONLY WITH THE HELP OF WRAPPER CLASS.