Day 10

String

```
* Members of String
        * Constructor
                1. public String()
                        - String s1 = new String();
                2. public String(byte[] bytes);
                        byte[] bytes = new byte[ ]{ 65, 66, 67 };
                        String s2 = new String( bytes );
                3. public String(char[] value)
                        char[] arr = new char[ ]{'A','B','C'};
                        String s3 = new String( arr );
                4. public String(String original)
                        String str = "SunBeam";
                        String s4 = new String( str );
                5. public String(StringBuffer buffer)
                        StringBuffer sb = new StringBuffer("CDAC");
                        String s5 = new String( sb );
                6. public String(StringBuilder builder)
                        StringBuilder sb = new StringBuilder("DMC");
                        String s6 = new String( sb );
*       String str = "ABC"
                is equivalent to:
        char[] data = { 'A', 'B', 'C'};
        String str = new String( data );
* Methods of String class:
        1. public char charAt(int index)
        2. public int codePointAt(int index)
        3. public int compareToIgnoreCase(String str)
        4. public String concat(String str)
        5. public boolean contains(CharSequence s)
        6. public boolean startsWith(String prefix)
        7. public boolean endsWith(String suffix)
        8. public boolean equalsIgnoreCase(String anotherString)
        9. public static String format(String format, Object... a);
        10. public byte[] getBytes();
        11. public int indexOf(int ch)
        12. public int indexOf(String str)
        13. public int lastIndexOf(int ch)
        14. public int lastIndexOf(String str)
        15. public int length()
        16. public boolean matches(String regex)
        17. public String[] split(String regex)
        18. public String substring(int beginIndex)
        19. public String substring(int beginIndex, int endIndex)
        20. public char[] toCharArray()
```

```
        21. public String toLowerCase()
        22. public String toUpperCase()
        23. public String trim()
        24. public static String valueOf(int i)
        25. public boolean isEmpty()
        26. public String intern()
* using regular expression, if we want to split string then we should use
"String[] split(String regex )" method.
```

Enumeration

```
* It is a interface declared in java.util package.
* Methods of Enumeration I/F:
        1. boolean hasMoreElements()
        2. E nextElement()
* It used to traverse collection in forward direction only. During
traversing, using enumeration, we can not add, set or remove element from
underlying collection.
```

StringTokenizer

```
* It is sub class of java.lang.Object class and it implements
java.util.Enumeration interface.
* On the basis of delimeter, if we want to split string then we should use
java.util.StringTokenizer class.
* Constructors of StringTokenizer
        1. public StringTokenizer(String str);
        2. public StringTokenizer(String str, String delim);
        3. public StringTokenizer(String str, String delim,
       boolean returnDelims);
* Methods of StringTokenizer
        1. Object class contain 11 methods
        2. Enumeration contain 2 mthods
        3. Following are StringTokenizer methods:
                - public int countTokens()
                - public boolean hasMoreTokens()
                - public String nextToken()
                - public String nextToken(String delim)
```

```
        public static void main(String[] args)
        {
                String str = "SunBeam Infotech Pune";
                String subStr =  str.substring(8, 16);
                System.out.println(subStr);     //Infotech
        }
```

```
	public static void main13(String[] args)
	{
		String str = "SunBeam Infotech Pune";
		String subStr =  str.substring(8);
		System.out.println(subStr);	//Infotech Pune
	}
	public static void main12(String[] args)
	{
		String str = "ab+bc*cd-de/ef";
		String deilm = "+*-/";
		StringTokenizer stk = new StringTokenizer(str,deilm,
true);
		while( stk.hasMoreTokens())
		{
			String token = stk.nextToken();
			System.out.println(token);
		}
	}
	public static void main11(String[] args)
	{
		String str = "ab+bc*cd-de/ef";
		String deilm = "+*-/";
		StringTokenizer stk = new StringTokenizer(str,deilm);
		while( stk.hasMoreTokens())
		{
			String token = stk.nextToken();
			System.out.println(token);
		}
	}
	public static void main10(String[] args)
	{
		String str = "www.gmail.com";
		String deilm = ".";
		StringTokenizer stk = new StringTokenizer(str,deilm);
		while( stk.hasMoreTokens())
		{
			String token = stk.nextToken();
			System.out.println(token);
		}
	}
	public static void main9(String[] args)
	{
		String str = "SunBeam Infotech Pune";
		StringTokenizer stk = new StringTokenizer(str);
		while( stk.hasMoreTokens())
		{
			String token = stk.nextToken();
			System.out.println(token);
		}
	}
	public static void main8(String[] args)
	{
		String str = "SunBeam Infotech Pune";
		StringTokenizer stk = new StringTokenizer(str);
```

```
                while( stk.hasMoreElements())
                {
                        String token = (String) stk.nextElement();
                        System.out.println(token);
                }
        }
        public static void main7(String[] args)
        {
                String str = "SunBeam Infotech Pune";
                StringTokenizer stk = new StringTokenizer(str);
                System.out.println(stk.countTokens());  //3
        }
        public static void main6(String[] args)
        {
                String regex = "\\.";
                String str = "www.yahoo.com";
                String[] words = str.split(regex);
                for (String word : words)
                {
                        System.out.println(word);
                }
        }
        public static void main5(String[] args)
        {
                String regex = " ";
                String str = "SunBeam Infotech Pune";
                String[] words = str.split(regex);
                for (String word : words)
                {
                        System.out.println(word);
                }
        }
        public static void main4(String[] args)
        {
                String s1 = "SunBeam";
                String s2 = "sunbeam";
                if( s1.equalsIgnoreCase(s2))
                        System.out.println("Equal");
                else
                        System.out.println("Not Equal");
                //Output : Equal
        }
        public static void main3(String[] args)
        {
                String s1 = "SunBeam";
                String s2 = "sunbeam";
                if( s1.equals(s2))
                        System.out.println("Equal");
                else
                        System.out.println("Not Equal");
                //Output : Not Equal
        }
        public static void main2(String[] args)
        {
```

```
                String str = "SunBeam";
                System.out.println("Length      :       "+str.length());
//7
        }
        public static void main1(String[] args)
        {
                String str = "Java";
                if( str.isEmpty())
                {
                        str = "Core Java";
                        System.out.println(str);
                }
                else
                        System.out.println(str);
        }
```

String Twisters:

```
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
}
public static void main10(String[] args)
{
        String s1 = "SunBeam";
        String str = "Sun";
        String s2 = str + "Beam";
        if( s1.equals(s2) )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main9(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 static void main8(String[] args)
{
        String s1 = "SunBeam";
        String s2 = "Sun"+"Beam";       //"SunBeam"
        if( s1.equals(s2))
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main7(String[] args)
{
        String s1 = "SunBeam";
        String s2 = "Sun"+"Beam";       //"SunBeam"
        if( s1 == s2 )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main6(String[] args)
{
        String s1 = "CDAC";
        String s2 = new String("CDAC");
        if( s1.equals(s2) )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main5(String[] args)
{
        String s1 = "CDAC";
        String s2 = new String("CDAC");
        if( s1 == s2 )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Not Equal
}
public static void main4(String[] args)
{
        String s1 = "CDAC";
        String s2 = "CDAC";
        if( s1.equals(s2))
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main3(String[] args)
{
        String s1 = "CDAC";
```

```
        String s2 = "CDAC";
        if( s1 == s2 )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main2(String[] args)
{
        String s1 = new String("CDAC");
        String s2 = new String("CDAC");
        if( s1.equals(s2))
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Equal
}
public static void main1(String[] args)
{
        String s1 = new String("CDAC");
        String s2 = new String("CDAC");
        if( s1 == s2 )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Not Equal
}
```

```
* Constant expression gets evaluated at compile time.
        int x = 2 + 3;
        //int x = 5;    //at compile time
* whereas non constant expression get evaluated at runtime.
        int x = 2, y = 3;
        int z = x + y;  //at runtime
* In case of non constant expression, if we want to return reference of
string from string literal pool then we should use intern() method.
```

```
import p1.A;
class B
{
        public static final String str = "SunBeam";
}
public class Program
{
        public static final String str = "SunBeam";
        public static void main(String[] args)
        {
                String str = "SunBeam";
                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 and StringBuilder

```
* Both are final classes declared in java.lang package
* If we want to create mutable string instances then we should create
instance of these classes.
* If we want to create instance of these classes then it is mandatory to
use new operator.
* Even though, these classes are final, "equals()" and "hashcode()" method
is not overridden in it.
```

> What is the difference between StringBuffer and StringBuilder? * StringBuffer implementation is synchronized(thread safe) whereas implementation of StringBuilder is unsynchronized.

```
public static void main(String[] args)
{
        try( Scanner sc = new Scanner(System.in))
        {
                System.out.print("Enter number  :       ");
                int num1 = sc.nextInt();
                System.out.println("Num1        :       "+num1);
                String strNum1 = String.valueOf(num1);
                StringBuilder sb = new StringBuilder(strNum1);
                sb.reverse();
                String strNum2 =  sb.toString();
                int num2 = Integer.parseInt(strNum2);
                System.out.println("Num2        :       "+num2);
        }
}
public static void main4(String[] args)
{
        StringBuilder sb1 = new StringBuilder("Pune");
        StringBuilder sb2 = new StringBuilder("Pune");
        if( sb1.equals(sb2) )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Not Equal
}
public static void main3(String[] args)
```

```
{
        StringBuilder sb1 = new StringBuilder("Pune");
        StringBuilder sb2 = new StringBuilder("Pune");
        if( sb1 == sb2 )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Not Equal
}
public static void main2(String[] args)
{
        StringBuffer sb1 = new StringBuffer("Sandeep");
        StringBuffer sb2 = new StringBuffer("Sandeep");
        if( sb1.equals(sb2))
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Not Equal
}
public static void main1(String[] args)
{
        StringBuffer sb1 = new StringBuffer("Sandeep");
        StringBuffer sb2 = new StringBuffer("Sandeep");
        if( sb1 == sb2 )
                System.out.println("Equal");
        else
                System.out.println("Not Equal");
        //Output : Not Equal
}
```

Collection Framework

```
* Data structure class is also called as collection.
* Framwork is a library of readymade classes that is used to develop
application.
* Collection framework is a library of readymade data structure/algorithm
classes that is used to develop efficient java application
* In java, collection instance do not contain instance rather it contains
reference of instance.
* To use collection framework, we must import java.util package.
```

Collection Framework Interface Hierarchy

```
- java.lang.Iterable<T>
        - java.util.Collection<E>
                - java.util.List<E>
                - java.util.Queue<E>
                        - java.util.Deque<E>
```

```
                - java.util.Set<E>
                        - java.util.SortedSet<E>
                                - java.util.NavigableSet<E>
```

Iterable

```
- It is interface declared in java.lang package.
- If we want traverse elements of any instance using foreach loop then
it's type must implement java.lang.Iterable<T> interface.
- Methods:
        1. Iterator<T> iterator()
        2. default Spliterator<T> spliterator()
        3. default void forEach(Consumer<? super T> action)
```

Collection

```
- It is a interface declared in java.util package.
- It is considered as root interface in collection hierarchy.
- Every object/value stored in collection instance is called     element.
- Abstract Methods of java.util.Collection interface.
        1. boolean add(E e)
        2. boolean addAll(Collection<? extends E> c)
        3. void clear()
        4. boolean contains(Object o)
        5. boolean containsAll(Collection<?> c)
        6. boolean remove(Object o)
        7. boolean removeAll(Collection<?> c)
        8. boolean retainAll(Collection<?> c)
        9. boolean isEmpty()
        10. int size()
        11. Object[] toArray()
        12. <T> T[] toArray(T[] a)
- Default Methods of java.util.Collection interface.
        1. default Stream<E> stream()
        2. default Stream<E> parallelStream()
        3. default boolean removeIf(Predicate<? super E> filter)
```

List

```
- It is sub interface of java.util.Collection<E>.
- ArrayList<E>, Vector<E>, Stack<E>, LinkedList<E> etc.         implements
List<E> interface. These are called as List collections.
- List collections are ordered/sequential collections.
- List collections can contain duplicate / multiple elements and null
```

```
values.
- We can access elements of List collections using integer index.
- We can traverse elements of List collection using Iterator as well as
ListIterator.
- If we want to manage, elements of non final type inside list collection
then non final type should override "equals()" method.
- Abstract Methods of java.util.List<E> interface
        1. void add(int index, E element)
        2. boolean addAll(int index, Collection<? extends E> c)
        3. E get(int index)
        4. int indexOf(Object o)
        5. int lastIndexOf(Object o)
        6. ListIterator<E> listIterator()
        7. ListIterator<E> listIterator(int index)
        8. E remove(int index)
        9. E set(int index, E element)
        10. List<E> subList(int fromIndex, int toIndex)

- Default Methods of java.util.List<E> interface
        1. default void sort(Comparator<? super E> c)
        2. default void replaceAll(UnaryOperator<E> operator)
```

ArrayList

```
- It is non final & concrete class declared in java.util package.
- It implements Following interfaces:
        1. java.util.List<E>
        2. java.util.RandomAccess
        3. java.lang.Cloneable
        4. java.io.Serializable
- It is resizable array.
- It is unsynchronized collection. If we want to make it synchronized then
we shoud use "Collections.synchronizedList" method.
List list = Collections.synchronizedList(new ArrayList(...));
- It is List<E> collection.
- By default, ArrayList can contain 10 elements. If it full then its
capacity gets increased by half of its existing capacity.
- It is introduced in jdk 1.2.
- If we want to manage, elements of non final type inside ArrayList<E>
then non final type should override "equals()" method.
- Constructor's of ArrayList :
        1. public ArrayList()
        2. public ArrayList(int initialCapacity)
        3. public ArrayList(Collection<? extends E> c)
- Method's of ArrayList
        1. public void ensureCapacity(int minCapacity)
        2. protected void removeRange(int fromIndex, int toIndex)
        3. public void trimToSize()
```

ArrayList Instantiation

```
public static void main(String[] args)
{
        //Collection<Integer> c = new ArrayList<>( );
        //Collection<Integer> c = new LinkedList<>( );
        //List<Integer> c = new ArrayList<>( );
        ArrayList<Integer> c = new ArrayList<>( );
        c.add(10);
        c.add(20);
        c.add(30);

        List<Integer> list2 = new ArrayList< >( c );
        System.out.println(list2.size());
}
public static void main2(String[] args)
{
        ArrayList<Integer> list1 = new ArrayList< >( 5 );       //OK
        List<Integer> list2 = new ArrayList< >( 7 );    //OK : Upcasting
        Collection<Integer> list3 = new ArrayList< >( 15 );     //OK :
Upcasting
}
public static void main1(String[] args)
{
        ArrayList<Integer> list1 = new ArrayList< >();  //OK
        List<Integer> list2 = new ArrayList< >();       //OK : Upcasting
        Collection<Integer> list3 = new ArrayList< >(); //OK : Upcasting
}
```

```
- Using illegal index, if we try to access elements of array(Single
dimensional, Multi dimensional and Ragged array ) then JVM throws
ArrayIndexOutOfBoundsException.
- Using illegal index, if we try to access character string then string
method throws StringIndexOutOfBoundsException.
- Using illegal index, if we try to find out element from List collection
then list methods throws IndexOutOfBoundsException.
```

```
public static void main(String[] args)
{
        List<Integer> list  = Program.getIntegerList( );
        int element = list.get( list.size() );
//IndexOutOfBoundsException
        System.out.println(element);
}
```

```
- IndexOutOfBoundsException is super class of
ArrayIndexOutOfBoundsException and StringIndexOutOfBoundsException.
```