StringBuilder

StringBuilder Constructors

- StringBuilder defines these four constructors:
  - StringBuilder( )
  - StringBuilder(int cap)
  - StringBuilder(String str)
- **The default constructor reserves room for 16 characters without reallocation.**
- By allocating room for a few extra characters(size +16) initially, StringBuilder reduces the number of reallocations that take place.

StringBuilder Methods

length( ) and capacity( )

The current length of a StringBuilder can be found via the length( ) method, while the total allocated capacity can be found through the capacity( ) method.

int length( )

int capacity( )

Example:

```
class StringBuilderDemo {
        public static void main(String args[]) {
                StringBuilder sb = new StringBuilder(“New Zealand");
                System.out.println("length = " + sb.length());
                System.out.println("capacity = " + sb.capacity());
          }
        }
```

ensureCapacity( )

- If we want to preallocate room for a certain number of characters after a StringBuilder has been constructed, we can use ensureCapacity( ) to set the size of the buffer.

- This is useful if we know in advance that we will be appending a large number of small strings to a StringBuilder.

void ensureCapacity(int capacity)

- Here, capacity specifies the size of the buffer.

Important

- When the length of StringBuilder becomes larger than the capacity then memory reallocation is done:

- In case of StringBuilder, reallocation of memory is done using the following rule:

- If the new_length <= 2*(original_capacity + 1), then

  new_capacity = 2*(original_capacity + 1)

- Else, new_capacity = new_length.

6) StringBuilder capacity() method

The capacity() method of StringBuilder class returns the current capacity of the Builder. The default capacity of the Builder is 16. If the number of character increases from its current capacity, it increases the capacity by (oldcapacity*2)+2. For example if your current capacity is 16, it will be (16*2)+2=34.

```
class A{
public static void main(String args[]){
StringBuilder sb=new StringBuilder();
System.out.println(sb.capacity());//default 16
sb.append("Hello");
System.out.println(sb.capacity());//now 16
sb.append("java is my favourite language");
System.out.println(sb.capacity());//now (16*2)+2=34 i.e (oldcapacity*2)+2
}
}
```

7) StringBuilder ensureCapacity() method

The ensureCapacity() method of StringBuilder class ensures that the given capacity is the minimum to the current capacity. If it is greater than the current capacity, it increases the capacity by (oldcapacity*2)+2. For example if your current capacity is 16, it will be (16*2)+2=34.

```
class A{
public static void main(String args[]){
StringBuilder sb=new StringBuilder();
System.out.println(sb.capacity());//default 16
sb.append("Hello");
System.out.println(sb.capacity());//now 16
sb.append("java is my favourite language");
System.out.println(sb.capacity());//now (16*2)+2=34 i.e (oldcapacity*2)+2
sb.ensureCapacity(10);//now no change
System.out.println(sb.capacity());//now 34
sb.ensureCapacity(50);//now (34*2)+2
System.out.println(sb.capacity());//now 70
}
}
```

Brain Storming

```
StringBuilder sb = new StringBuilder();
System.out.println(sb.capacity());


StringBuilder sb = new StringBuilder(65);
System.out.println(sb.capacity());


StringBuilder sb = new StringBuilder(“A”);
System.out.println(sb.capacity());


StringBuilder sb = new StringBuilder('A');
System.out.println(sb.capacity());
```

Brainstorming

What will be the output?

```
class DemoString
{
        public static void main(String...rk)
        {
                StringBuilder b = new StringBuilder("abcdef");
                //b.delete(4,6);
                b.ensureCapacity(22);
                System.out.print(b.capacity());
                b.ensureCapacity(23);
                System.out.print(b.capacity());
        }
}
```

setLength( )

- used to set the length of the buffer within a StringBuilder object.

  void setLength(int length)

- Here, length specifies the length of the buffer.

- When we increase the size of the buffer, null characters are added to the end of the existing buffer.

- If we call setLength( ) with a value less than the current value returned by length( ), then the characters stored beyond the new length will be lost.

charAt( ) and setCharAt( )

- The value of a single character can be obtained from a StringBuilder via the charAt( ) method.

- We can set the value of a character within a StringBuilder using setCharAt( ).

char charAt(int index)

void setCharAt(int index, char ch)

getChars( )

- getChars( ) method is used to copy a substring of a StringBuilder into an array.

void getChars(int sourceStart,   int sourceEnd,   char target[ ],   int targetStart)

append( )

- The append( ) method concatenates the string representation of any other type of data to the end of the invoking StringBuilder object.

- It has several overloaded versions.
  - StringBuilder append(String str)
  - StringBuilder append(int num)
  - StringBuilder append(Object obj)

Example

```
class appendDemo
{
        public static void main(String args[])
        {
                String s;
                int a = 42;
                StringBuilder sb = new StringBuilder(40);
                s = sb.append("a = ").append(a).append("!").toString();
                System.out.println(s);
        }
}
```

insert( )

- The insert( ) method inserts one string into another.
- It is overloaded to accept values of all the simple types, plus Strings, Objects, and CharSequences.
- This string is then inserted into the invoking StringBuilder object.
    - StringBuilder insert(int index, String str)
    - StringBuilder insert(int index, char ch)
    - StringBuilder insert(int index, Object obj)

Example

```
class insertDemo
{
        public static void main(String args[])
        {
                StringBuilder sb = new StringBuilder("I Java!");
                sb.insert(2, “don’t like ");
                System.out.println(sb);
        }
}
```

reverse( )

- Used to reverse the characters within a StringBuilder object.
- This method returns the reversed object on which it was called.

StringBuilder reverse()

Example:

```
class ReverseDemo
{
        public static void main(String args[])
        {
                StringBuilder s = new StringBuilder(“Banana");
                System.out.println(s);
                s.reverse();
                System.out.println(s);
        }
}
```

Exercise

Write a program which prompts the user to enter a String.

- Check Whether the String is Palindrome or Not?

delete( ) and deleteCharAt( )

- Used to delete characters within a StringBuilder.

StringBuilder delete(int startIndex, int endIndex)

StringBuilder deleteCharAt(int index)

- The delete( ) method deletes a sequence of characters from the invoking object (from startIndex to endIndex-1).

- The deleteCharAt( ) method deletes the character at the specified index.

- It returns the resulting StringBuilder object.

Example

```
class deleteDemo
{
        public static void main(String args[])
        {
                StringBuilder sb = new StringBuilder(“She is not a good girl.”);
                sb.delete(7, 11);
                System.out.println("After delete: " + sb);
                sb.deleteCharAt(7);
                System.out.println("After deleteCharAt: " + sb);
        }
}
```

replace( )

- Used to replace one set of characters with another set inside a StringBuilder object.

StringBuilder replace(int startIndex, int endIndex, String str)

- The substring being replaced is specified by the indexes startIndex and endIndex.
- Thus, the substring at startIndex through endIndex–1 is replaced.
- The replacement string is passed in str.
- The resulting StringBuilder object is returned.

Example

```
class replaceDemo
{
        public static void main(String args[])
        {
            StringBuilder sb = new StringBuilder("This is a test.");
            sb.replace(5, 7, "was");
            System.out.println("After replace: " + sb);
        }
}
```

substring( )

- Used to obtain a portion of a StringBuilder by calling substring( ).

String substring(int startIndex)

String substring(int startIndex, int endIndex)

- The first form returns the substring that starts at startIndex and runs to the end of the invoking StringBuilder object.

- The second form returns the substring that starts at startIndex and runs through endIndex–1.