provide all the string methods of the string buffer which are not same as string methods in java with core examples

Excellent question, Routh 👏 — this is a **key concept** when learning how `StringBuffer` differs from `String` in Java.

Let's go step-by-step and cover **all methods of `StringBuffer` that are NOT available in `String`**, along with **clear explanations and practical examples**.

🧠 Quick Recap — `String` vs `StringBuffer`

Feature	`String`	`StringBuffer`
Mutability	Immutable	Mutable
Thread Safety	Not synchronized	Synchronized
Performance	Slower for repeated modifications	Faster for modifications
Package	`java.lang`	`java.lang`

🧩 Methods unique to `StringBuffer`

These methods deal with **modifying**, **inserting**, **reversing**, or **managing capacity**, which `String` doesn't allow since it's immutable.

---

1. `append()`

Purpose: Adds text (or any data type) to the end of the buffer.

Syntax:

StringBuffer append(data);

Example:

```
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World");
System.out.println(sb); // Output: Hello World
```

👉 Not available in `String` because `String` is immutable — you'd use `str = str + " World"` instead.

---

2. `insert()`

Purpose: Inserts text at a specific position.

Syntax:

```
StringBuffer insert(int offset, data);
```

Example:

```
StringBuffer sb = new StringBuffer("Java");
sb.insert(2, "Cool");
System.out.println(sb); // Output: JaCoolva
```

---

3. `replace()`

Purpose: Replaces a range of characters with new text.

Syntax:

```
StringBuffer replace(int start, int end, String str);
```

Example:

```
StringBuffer sb = new StringBuffer("I love Python");
sb.replace(7, 13, "Java");
System.out.println(sb); // Output: I love Java
```

*(Note: `String` has `replace()` too, but it replaces characters or substrings, not a **range** of indices like this one.)*

---

4. `delete()`

Purpose: Deletes characters from start index to end index.

Syntax:

StringBuffer delete(int start, int end);

Example:

StringBuffer sb = new StringBuffer("Programming");
sb.delete(3, 6);
System.out.println(sb); // Output: Proamming

---

5. `deleteCharAt()`

Purpose: Deletes a single character at the specified index.

Syntax:

StringBuffer deleteCharAt(int index);

Example:

StringBuffer sb = new StringBuffer("Hello");
sb.deleteCharAt(1);
System.out.println(sb); // Output: Hllo

---

6. `reverse()`

Purpose: Reverses the character sequence.

Syntax:

StringBuffer reverse();

Example:

```
StringBuffer sb = new StringBuffer("Java");
sb.reverse();
System.out.println(sb); // Output: avaJ
```

---

7. `capacity()`

Purpose: Returns the **current storage capacity** (number of characters it can hold before resizing).

Syntax:

```
int capacity();
```

Example:

```
StringBuffer sb = new StringBuffer();
System.out.println(sb.capacity()); // Default: 16
sb.append("Hello");
System.out.println(sb.capacity()); // Still 16 (no resize yet)
```

---

8. `ensureCapacity()`

Purpose: Ensures that the buffer can hold at least the specified number of characters.

Syntax:

```
void ensureCapacity(int minimumCapacity);
```

Example:

```
StringBuffer sb = new StringBuffer("Java");
sb.ensureCapacity(50);
System.out.println(sb.capacity()); // Output: >= 50
```

---

9. `setLength()`

Purpose: Sets the length of the buffer.
If new length is shorter → characters are truncated.
If longer → null characters (`\u0000`) are added.

Syntax:

```
void setLength(int newLength);
```

Example:

```
StringBuffer sb = new StringBuffer("HelloWorld");
sb.setLength(5);
System.out.println(sb); // Output: Hello
```

---

10. `trimToSize()`

Purpose: Reduces the capacity to match the current length (to save memory).

Syntax:

```
void trimToSize();
```

Example:

```
StringBuffer sb = new StringBuffer("Java");
sb.ensureCapacity(50);
sb.trimToSize();
System.out.println(sb.capacity()); // Output: 4
```

---

11. `setCharAt()`

Purpose: Changes a character at a specific index.

Syntax:

```
void setCharAt(int index, char ch);
```

Example:

```
StringBuffer sb = new StringBuffer("Rava");
sb.setCharAt(0, 'J');
System.out.println(sb); // Output: Java
```

🗒️ Summary Table

Method	Description	Available in `String`?
append()	Adds data at the end	❌
insert()	Inserts data at index	❌
replace(int,int,String)	Replace range	⚠️ (Different overload)
delete()	Delete range	❌
deleteCharAt()	Delete one char	❌
reverse()	Reverse string	❌
capacity()	Check current buffer capacity	❌
ensureCapacity()	Ensure min capacity	❌
setLength()	Change length manually	❌
trimToSize()	Reduce capacity to length	❌
setCharAt()	Modify a specific character	❌

✅ Bonus Tip

You can **convert between String and StringBuffer**:

```
StringBuffer sb = new StringBuffer("Hello");
String str = sb.toString(); // StringBuffer → String

StringBuffer sb2 = new StringBuffer(str); // String → StringBuffer
```

Would you like me to create a **small Java program** that demonstrates **all these methods together** (so you can run and observe the outputs step-by-step)?
It would be perfect for your Java learning notebook.